Commande par linéarisation entrée-sortie d'un drone de type quadcopter à l'aide de la Kinect One

par

Brice HERNANDEZ

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE ÉLECTRIQUE
M.Sc.A.

MONTRÉAL, LE 11 OCTOBRE 2017

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Maarouf Saad, directeur de mémoire
Département de génie électrique à l'École de Technologie Supérieure

M. Vahé Nerguizian, président du jury
Département de génie électrique à l'École de Technologie Supérieure

M. Jawhar Ghommam, membre du jury
Membre externe, Université de Carthage

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 4 OCTOBRE 2017

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

J'aimerais remercier dans un premier temps le professeur Maarouf Saad pour m'avoir permis d'accroitre mes connaissances en robotique et en logiciel en me donnant ce projet. Je remercie aussi le professeur Jawhar Ghommam pour m'avoir proposé ce sujet de maitrise. Je remercie notamment tous mes collègues du GREPCI pour leur soutien et encouragements notamment Walid Alqaisi et Nuradeen Fethalla.

Je remercie particulièrement Jean Marie Lauhic Ndong Mezui pour m'avoir supporté pendant des nuits de travail. Je remercie aussi Brahim Brahmi pour ses conseils et son soutien.

J'aimerais enfin remercier toute ma famille pour m'avoir encouragé depuis la France avec une pensée particulière pour mon père, qui, juste au bout de sa vie, n'a cessé de croire en moi et faire tout son possible pour que j'arrive à mes objectifs.

COMMANDE PAR LINÉARISATION ENTRÉE-SORTIE D'UN DRONE DE TYPE QUADCOPTER À L'AIDE DE LA KINECT ONE

Brice HERNANDEZ

RÉSUMÉ

Le sujet de ce mémoire se porte sur l'implémentation d'un contrôleur pour un quadcopter. Deux types de drones seront utilisés : Le Crazyflie v2.0 ainsi qu'un drone basé sur un châssis S500. Une modélisation du drone sera faite dans un premier temps. Deux contrôleurs par linéarisation entrée-sortie seront développés : un contrôleur d'attitude permettant le contrôle de l'orientation ainsi qu'un contrôleur de position. Une loi d'adaptation sera introduite dans le contrôleur de la position permettant de contrer des perturbations extérieures. La Kinect One sera utilisée pour récupérer la position actuelle du drone. On utilisera une soustraction de l'arrière-plan comme algorithme. Deux filtres seront présentés afin d'estimer la vitesse du drone : un filtre passe-bas du premier ordre ainsi qu'un filtre de Kalman. Deux méthodes seront traitées pour obtenir les moments d'inertie du drone. La première méthode se base sur l'approximation du drone en objet simple (cylindres et tiges) permettant de facilement récupérer les moments d'inertie. La deuxième méthode se basera sur le pendule tri filaire. Les deux méthodes permettent d'obtenir des résultats similaires. Le contrôleur sera implémenté à l'aide du micrologiciel Crazyflie Firmware pour le Crazyflie et avec le micrologiciel PX4 pour le drone S500. Le drone S500 sera muni d'un Pixhawk suivi d'un Odroid pour effectuer les expérimentations. Avec ces configurations, les drones suivent alors une trajectoire en cercle d'un rayon de 40 cm et de période de 40 secondes avec une erreur de position de moins de 10 cm pour le Crazyflie et une erreur de moins de 6 cm pour le drone S500 en régime permanent.

Mots clés: quadcopter, drone, linéarisation entrée-sortie, ROS, Crazyflie, Pixhawk, Odroid, Kalman, Kinect One

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

IMU	Inertial Measurement Unit
MLI	Modulation de Largeur d'Impulsion
ESC	Electronic Speed Control
AHRS	Attitude and Heading Reference System
I2C	Inter-Integrated Circuit
UART	Universal Asynchronous Receiver Transmitter
ROS	Robot Opereating System
MPI	Modulation en Position d'Impuslion
MAVLINK	Micro Air Vehicle Link
ULB	Ultra Large Bande
PID	Proportionnel Intégrateur Dérivé
LQR	Linear Quadratic Regulator
GPS	Global Positioning System
CRTP	Crazy Real Time Protocol

LISTE DES SYMBOLES ET UNITÉS DE MESURE

M_1	Moteur 1 du drone
M_2	Moteur 2 du drone
M_3	Moteur 3 du drone
M_4	Moteur 4 du drone
ω_1	Vitesse angulaire du moteur 1 ($\text{rad}\cdot\text{s}^{-1}$)
ω_2	Vitesse angulaire du moteur 2 ($\text{rad}\cdot\text{s}^{-1}$)
ω_3	Vitesse angulaire du moteur 3 ($\text{rad}\cdot\text{s}^{-1}$)
ω_4	Vitesse angulaire du moteur 4 ($\text{rad}\cdot\text{s}^{-1}$)
$\{i\}$	Repère inertiel
$\{b\}$	Repère du corps du drone
ϕ	Angle de roulis (rad)
θ	Angle de tangage (rad)
ψ	Angle de lacet (rad)
r_i	Premier repère auxillaire
r_ψ	Deuxième repère auxillaire
r_θ	Troisième repère auxillaire
${}^{r_\psi}_{r_i}R$	Matrice de rotation passant du repère r_ψ au repère r_i
${}^{r_\theta}_{r_\psi}R$	Matrice de rotation passant du repère r_θ au repère r_ψ
${}^{b}_{r_\theta}R$	Matrice de rotation passant du repère $\{b\}$ au repère r_θ
${}^{i}_{b}R$	Matrice de rotation passant du repère $\{i\}$ au repère $\{b\}$
${}^{b}\omega$	Vecteur des vitesses angulaires exprimées dans le repère $\{b\}$ ($\text{rad}\cdot\text{s}^{-1}$)
p	Vitesse angulaire exprimé dans le repère $\{b\}$ autour de l'axe x du repère $\{b\}$ ($\text{rad}\cdot\text{s}^{-1}$)

q	Vitesse angulaire exprimé dans le repère $\{b\}$ autour de l'axe y du repère $\{b\}$ ($\mathrm{rad \cdot s^{-1}}$)
r	Vitesse angulaire exprimé dans le repère $\{b\}$ autour de l'axe z du repère $\{b\}$ ($\mathrm{rad \cdot s^{-1}}$)
J	Matrice de rotation entre les vitesses angulaires et les vitesses des angles de rotation
η	Vecteur des angles de rotation (rad)
m	Masse du drone (kg)
X_i	Vecteur de position actuelle du drone exprimé dans le repère inertiel (m)
x_i	Position actuelle dans le repère inertiel du drone suivant l'axe x du repère $\{i\}$ (m)
y_i	Position actuelle dans le repère inertiel du drone suivant l'axe y du repère $\{i\}$ (m)
z_i	Position actuelle dans le repère inertiel du drone suivant l'axe z du repère $\{i\}$ (m)
g	Accélération gravitationnelle ($\mathrm{m \cdot s^{-2}}$)
Δ	Vecteur de la perturbation extérieure exprimé dans le repère inertiel ($\mathrm{m \cdot s^{-2}}$)
Δ_x	Perturbation extérieure exprimée dans le repère inertiel suivant l'axe x du repère $\{i\}$ ($\mathrm{m \cdot s^{-2}}$)
Δ_y	Perturbation extérieure exprimée dans le repère inertiel suivant l'axe y du repère $\{i\}$ ($\mathrm{m \cdot s^{-2}}$)
Δ_z	Perturbation extérieure exprimée dans le repère inertiel suivant l'axe z du repère $\{i\}$ ($\mathrm{m \cdot s^{-2}}$)
T	Force de portance (N)
${}^b v$	Vecteur des vitesses linéaire exprimé dans repère du corps ($\mathrm{m \cdot s^{-1}}$)
u	Vitesse linéaire exprimée dans repère du corps suivant l'axe x du repère $\{b\}$ ($\mathrm{m \cdot s^{-1}}$)

v	Vitesse linéaire exprimée dans repère du corps suivant l'axe y du repère $\{b\}$ ($\mathrm{m{\cdot}s^{-1}}$)
w	Vitesse linéaire exprimée dans repère du corps suivant l'axe z du repère $\{b\}$ ($\mathrm{m{\cdot}s^{-1}}$)
u_Θ	Vecteur du couple du drone exprimé dans le repère $\{b\}$ (Nm)
u_ϕ	Couple du drone exprimé dans le repère $\{b\}$ autour de l'axe x du repère $\{b\}$ (Nm)
u_θ	Couple du drone exprimé dans le repère $\{b\}$ autour del'axe y du repère $\{b\}$ (Nm)
u_ψ	Couple du drone exprimé dans le repère $\{b\}$ autour de l'axe z du repère $\{b\}$ (Nm)
τ_g	Vecteur du couple de l'effet gyroscopique (Nm)
J_m	Inertie du rotor d'un moteur du drone ($\mathrm{kg{\cdot}m^2}$)
Ω_r	Vitesse angulaire relative du drone ($\mathrm{rad{\cdot}s^{-1}}$)
I	Tenseur d'inertie du drone ($\mathrm{kg{\cdot}m^2}$)
I_{xx}	Moment d'inertie autour de l'axe x du repère $\{b\}$ ($\mathrm{kg{\cdot}m^2}$)
I_{yy}	Moment d'inertie autour de l'axe y du repère $\{b\}$ ($\mathrm{kg{\cdot}m^2}$)
I_{zz}	Moment d'inertie autour de l'axe z du repère $\{b\}$ ($\mathrm{kg{\cdot}m^2}$)
X	Variable d'état du drone
${}^iF$	Vecteur de force exprimé dans le repère inertiel (N)
${}^iF_x$	Force exprimée dans le repère inertiel suivant l'axe x du repère $\{i\}$ (N)
${}^iF_y$	Force exprimée dans le repère inertiel suivant l'axe y du repère $\{i\}$ (N)
${}^iF_z$	Force exprimée dans le repère inertiel suivant l'axe z du repère $\{i\}$ (N)
U	Vecteur de force-couple exprimé dans le repère inertiel
F_1	Force produite par le moteur M_1 (N)
F_2	Force produite par le moteur M_2 (N)

F_3	Force produite par le moteur M_3 (N)
F_4	Force produite par le moteur M_4 (N)
τ_1	Couple produit par le moteur M_1 (Nm)
τ_2	Couple produit par le moteur M_2 (Nm)
τ_3	Couple produit par le moteur M_3 (Nm)
τ_4	Couple produit par le moteur M_4 (Nm)
ρ	Densité de l'air ($kg \cdot m^{-3}$)
D	Diamètre d'un hélice (m)
$f(\omega_n)$	Fonction reliant le couple d'un moteur avec sa force
Θ	Variable d'état incluant les angles et les vitesses angulaires
X_{pos}	Variable d'état incluant les position du drone (m)
$k_{\Theta p}$	Gain proportionnel du contrôleur d'attitude ($rad \cdot s^{-2}$)
$k_{\Theta d}$	Gain dérivé du contrôleur d'attitude ($rad \cdot s^{-1}$)
Θ_d	Vecteur des angles désirés (rad)
ϕ_d	Angle de roulis désiré (rad)
θ_d	Angle de tangage désiré (rad)
ψ_d	Angle de lacet désiré (rad)
v	Vecteur de la loi de commande pour le contrôleur d'attitude ($rad \cdot s^{-2}$)
v_ϕ	Loi de commande pour l'accéleration angulaire de l'angle ϕ ($rad \cdot s^{-2}$)
v_θ	Loi de commande pour l'accéleration angulaire de l'angle θ ($rad \cdot s^{-2}$)
v_ψ	Loi de commande pour l'accéleration angulaire de l'angle ψ ($rad \cdot s^{-2}$)
e_ϕ	Erreur entre l'angle désiré ϕ_d et l'angle ϕ (rad)
e_θ	Erreur entre l'angle désiré θ et l'angle θ (rad)
e_ψ	Erreur entre l'angle désiré ψ_d et l'angle ψ (rad)

x_{id}	Position désirée dans le repère inertiel du drone suivant l'axe x du repère $\{i\}$ (m)
y_{id}	Position désirée dans le repère inertiel du drone suivant l'axe y du repère $\{i\}$ (m)
z_{id}	Position désirée dans le repère inertiel du drone suivant l'axe z du repère $\{i\}$ (m
e_x	Erreur entre la position x_{id} et la position x_i (m)
e_y	Erreur entre la position y_{id} et la position y_i (m)
e_z	Erreur entre la position z_{id} et la position z_i (m)
k_p	Gain proportionnel du contrôleur de position (s^{-2})
k_d	Gain dérivé du contrôleur de position (s^{-1})
$\hat{\Delta}_x$	Perturbation extérieure éstimée exprimée dans le repère inertiel suivant l'axe x du repère $\{i\}$ ($m{\cdot}s^{-2}$)
$\hat{\Delta}_y$	Perturbation extérieure éstimée exprimée dans le repère inertiel suivant l'axe y du repère $\{i\}$ ($m{\cdot}s^{-2}$)
$\hat{\Delta}_z$	Perturbation extérieure éstimée exprimée dans le repère inertiel suivant l'axe z du repère $\{i\}$ ($m{\cdot}s^{-2}$)
$\tilde{\Delta}_x$	Erreur d'estimation entre Δ_x et $\hat{\Delta}_x$ ($m{\cdot}s^{-2}$)
$\tilde{\Delta}_y$	Erreur d'estimation entre Δ_y et $\hat{\Delta}_y$ ($m{\cdot}s^{-2}$)
$\tilde{\Delta}_z$	Erreur d'estimation entre Δ_z et $\hat{\Delta}_z$ ($m{\cdot}s^{-2}$)
Γ_x	Gain de la loi d'adaptation pour $\tilde{\Delta}_x$ (s^{-1})
Γ_y	Gain de la loi d'adaptation pour $\tilde{\Delta}_y$ (s^{-1})
Γ_z	Gain de la loi d'adaptation pour $\tilde{\Delta}_z$ (s^{-1})
$\{f\}$	Repère rétinien
$\{k\}$	Repère du plan de l'image en millimètre
$\{c\}$	Repère du plan de l'image en pixel

${}^{c}_{k}T$	Matrice de transformation passant du repère $\{k\}$ à $\{c\}$
$\{f_0\}$	Distance focale (mm)
u_c	Axe suivant x de l'image en pixel
v_c	Axe suivant -y de l'image en pixel
x_{kdrone}	Projection de la position x_i sur le repère $\{k\}$ (mm)
y_{kdrone}	Projection de la position y_i sur le repère $\{k\}$ (mm)
x_{fdrone}	Projection de la position x_i sur le repère $\{f\}$ (mm)
y_{fdrone}	Projection de la position y_i sur le repère $\{f\}$ (mm)
z_{fdrone}	Projection de la position z_i sur le repère $\{f\}$ (mm)
k_u	Gain permettant de faire la relation entre la distance en pixel et en millimètre dans la direction u_c (pixel·s^{-1})
v_u	Gain permettant de faire la relation entre la distance en pixel et en millimètre dans la direction v_c (pixel·s^{-1})
θ_u	Angle du repère $\{c\}$ (rad)
u_{raw}	Signal bruité en entrée filtre (m)
y_{raw}	Signal de sortie du filtre (m)
ω_r	Pulsation du filtre passe-bas (rad·s)
ζ	Taux d'amortissement du filtre passe-bas
B_0	Bloqueur d'ordre 0 (S)
$\hat{x}_i$	Estimation de la position x_i (m)
$\hat{y}_i$	Estimation de la position y_i (m)
$\hat{z}_i$	Estimation de la position z_i (m)
T_{ech}	Temps d'echantillonnage (s)
w_{kal}	Bruit blanc venant du procédé (mm)
v_{kal}	Bruit blanc venant de la mesure (mm)

σ_x	Ecart-type du bruit de la position x_i (mm)
σ_y	Ecart-type du bruit de la position y_i (mm)
σ_z	Ecart-type du bruit de la position z_i (mm)
Q_{kal}	Matrice de covariance pour v_{kal} (mm^2)
R_{kal}	Matrice de covariance pour w_{kal} (mm^2)
m_{moteur}	Masse d'un moteur du drone (kg)
m_{base}	Masse du chassis du drone (kg)
m_{bras}	Masse d'un bras du drone (kg)
m_{ref}	Masse de l'objet de référence (kg)
l	Distance entre le centre du moteur et le centre du chassis du drone (m)
l_{bras}	Longueur d'un bras du drone (m)
r_{base}	Rayon du chassis du drone (m)
r_{disque}	Rayon du disque d'essai (m)
I_{disque}	Inertie du disque d'essai ($kg \cdot m^2$)
I_{ref}	Inertie de l'objet de référence ($kg \cdot m^2$)
l_{fils}	Longueur du fils du disque d'essai(m)
τ_{disque}	Periode d'oscillation du disque (s)
τ_{ref}	Periode d'oscillation de l'objet de référence (s)
$V_{batterie}$	Tension de la batterie du drone (V)

INTRODUCTION

La recherche sur les drones n'a cessé d'augmenter au cours de ces dernières décennies. Les drones ont progressivement été ouverts au grand public permettant d'être utilisés dans plusieurs domaines. On retrouve les drones dans le domaine médical, audiovisuel, de défense et de sécurité. Les drones commencent aussi à être répandus dans le domaine de la livraison notamment avec les achats en ligne.

Les drones, de type à voiture tournante, disposent de 6 degrés de liberté (où deux degrés de liberté peuvent se perdre sous certaines conditions). Leurs tailles varient du centimètre au mètre. La dynamique du drone peut être modélisée facilement permettant l'implémentation de plusieurs types de commande. Les composants de drones deviennent aussi de plus en plus bon marché. Ces propriétés sont quelques raisons pour l'expansion des recherches sur les drones.

Les drones sont souvent utilisés à l'extérieur à l'aide d'un GPS (Global Positioning System). La plupart des recherches se passent cependant dans des laboratoires. Le GPS ne pouvant pas être utilisé, un autre système de détection de position doit être trouvé. Les solutions les plus répandues sont alors les capteurs des mouvements. Ces capteurs peuvent s'avérer cependant coûteux, limitant le développement de la recherche sur les drones.

La Kinect de Microsoft, destinée, dans un premier temps, pour le divertissement, a fait l'objet de plusieurs recherches sur la détection des mouvements humains. Ces recherches se sont étendues sur la détection des objets. Le prix de la Kinect étant abordable, l'intérêt d'utiliser la caméra est devenu important. En 2013, Microsoft sort une version améliorée de la Kinect appelée Kinect One. Cette caméra offre une meilleure précision et une caméra haute définition. Utiliser la Kinect One pour détecter la position d'un drone permet alors de tester des commandes dans des laboratoires à un prix d'équipement abordable.

L'objectif de ce mémoire est de développer un contrôleur pour le drone en prenant en compte la Kinect One. Les essais se feront dans un laboratoire à l'intérieur dans une surface de quatre mètres carrés au maximum. Les expérimentations se feront dans un enclos. Seul le drone sera dans le champ de vision de la Kinect. Deux types de drones seront expérimentés. On modélisera dans un premier temps la dynamique du drone. On utilisera ensuite cette modélisation afin de concevoir un contrôleur en position et en attitude en utilisant une linéarisation entrée-sortie. On développera, par la suite, un algorithme se basant de la soustraction de l'arrière plan permettant de récupérer, à partir de la Kinect, la position du drone. On estimera ensuite la vitesse actuelle du drone par un filtre passe-bas dans un premier temps ainsi qu'un filtre de Kalman. Trois méthodes seront alors proposées afin de récupérer les paramètres du moteur. On pourra enfin valider la commande par la simulation et par l'expérimentation en utilisant un micro drone appelé Crazyflie ainsi que le drone S500 equipé d'une carte électronique appelée Pixhawk.

Ce mémoire est composé de 6 chapitres. Le premier chapitre présente une revue de littérature sur les drones. Le deuxième chapitre montre une modélisation du drone. Le troisième chapitre décrit trois méthodes différentes afin de relever les paramètres du drone. Le quatrième chapitre présente une loi de commande ainsi qu'une loi d'adaptation pour le drone. Le cinquième chapitre propose deux méthodes afin de relever la vitesse du drone ainsi que la position. Le sixième chapitre présente les résultats en simulation ainsi que des résultats expérimentaux de la commande.

CHAPITRE 1

REVUE DE LITTÉRATURE

1.1 Introduction

Les drones ont évolué tout au long de ce siècle pour être utilisés aujourd'hui dans plusieurs domaines. Le drone est un engin volant n'ayant pas recours à un pilote humain. Ces aéronefs autonomes ont été utilisés pour la première fois dans le domaine militaire pendant la Première Guerre mondiale. L' évolution de la technologie notamment au niveau de l' électronique, a permis d'améliorer les performances du drone de manière significative. Les drones sont à présent employés dans les domaines médical (Pulver *et al.* (2016)), audiovisuel (Nägeli *et al.* (2017)) et de l'ingénierie (McCabe *et al.* (2017)). Plusieurs types de drones existent. Les drones dits à voilure fixe (Figure 1.2) se comportent comme des avions et sont beaucoup utilisés dans le domaine militaire. Les drones dits à voilure tournante (Figure 1.1) se comportent comme des hélicoptères et sont beaucoup utilisés dans le domaine de l' audiovisuel et de la sécurité.

Figure 1.1 ASV 100
Tirée de Groizeleau (2014)

Figure 1.2 RQ-4 Global Hawk
Tirée de Mer et Marine (2014)

Ce chapitre évoquera une catégorie des drones appelés quadcopter. On notera, de manière non exhaustive, les différentes technologies derrière ces appareils, leurs applications, les différents contrôleurs utilisés pour piloter le drone ainsi que les différents composants nécessaires au fonctionnement du quadcopter.

1.2 Le quadcopter

Le quadcopter ainsi que les drones à voilure tournante se différencient aux drones à voilure fixe par leurs capacités à se déplacer avec un degré de liberté plus important. Le quadcopter a l'avantage d'être pilotable facilement et d'être bon marché. Sa capacité à pouvoir faire des vols stationnaires et à basses vitesses fait du quadcopter un élément intéressant dans l'utilisation du drone dans le domaine de la recherche. Ces drones peuvent avoir plusieurs tailles différentes, de la taille d'une pièce de monnaie (Figure 1.3) à plusieurs mètres de long. Les quadcopters de petites tailles sont souvent catégorisés en tant que micro drones et disposent de propriétés particulières. Le quadcopter fait partie de la catégorie des drones à voilure ayant plusieurs rotors. Il est composé de cinq éléments principaux : les moteurs au nombre de quatre, le châssis, la centrale inertielle (IMU), la carte électronique et le récepteur de radiocommunication.

Figure 1.3 Crazyflie v2
Tirée de RobotShop (2017)

1.3 Les moteurs utilisés

Les moteurs utilisés peuvent être de type avec ou sans balais. Les moteurs avec balais sont généralement utilisés pour les micro drones, leurs tailles pouvant être petites. Le moteur sans balais est facile à commander. Une modulation de largeur d'impulsion (MLI) est souvent utilisée pour contrôler les vitesses des moteurs. La vitesse du moteur est proportionnelle à la période cyclique du signal MLI (Hughes (2006b)) lorsque la tension d'alimentation est constante. Ce moteur a besoin d'une carte de puissance afin de fournir assez de courant aux rotors. Son rendement énergétique peut-être assez faible dû à sa structure lorsque l'on souhaite utiliser des vitesses élevées. On préfère alors utiliser les moteurs sans balais pour des drones plus imposants. Les moteurs sans balais utilisent la même structure qu'un moteur synchrone (Hughes (2006a)). On peut, à l'aide d'un variateur de vitesse (ESC), utiliser le moteur sans balais avec une tension continue comme alimentation. Le variateur de vitesse génère alors trois signaux déphasés de $\frac{2\pi}{3}$ radians pouvant être de forme trapézoïdale ou sinusoïdale aux phases du moteur. Le variateur de vitesse demande en général en entrée un signal MLI périodique de fréquence comprise entre 50 et 400 Hertz d'une largeur d'impulsion entre 1000 et 2000 microsecondes.

1.4 Le centre inertiel

Le centre inertiel est une des composantes les plus importantes du drone. Il permet de récupérer les angles d'orientation ainsi que les vitesses angulaires. Le centre inertiel contient trois capteurs ou deux capteurs : un gyroscope, un accéléromètre et un magnétomètre si l'on parle d'un centre inertiel à 9 degrés de liberté. Le magnétomètre n'est pas inclus lorsque l'on parle d'un centre inertiel à 6 degrés de liberté. Le gyroscope mesure la vitesse angulaire du drone. Ce capteur a l'avantage de ne pas être trop bruité et n'est pas influencé par le champ électromagnétique. L'intégration directe pour avoir les angles d'orientation est déconseillée. L'intégration cumule en effet les erreurs au cours du temps. Un décalage entre l'angle actuel et l'angle mesuré se fera au cours du temps. L' accéléromètre mesure l'accélération linéaire exercée par le drone. Ce capteur mesure correctement l'accélération si le dispositif ne vibre pas fortement. Les accéléromètres sont très sensibles aux forces extérieures ce qui peut causer des erreurs d'interpolation (Woodman (2007)). Ils ne peuvent donc pas être utilisés pour récupérer directement la position en intégrant deux fois ou être utilisés uniquement pour trouver les angles d'orientation.

Les capteurs ne peuvent individuellement mesurer les angles d'orientation du drone. Une solution proposée dans la littérature est de faire une fusion de capteur (*sensor fusion* en anglais). Cette méthode permet de combiner les informations données par les deux capteurs en même temps afin d'avoir un résultat plus précis sur les angles d'orientation. La fusion de capteur peut se faire de différentes manières. L'utilisation du filtre complémentaire a été utilisée dans de nombreuses recherches (Euston *et al.* (2008) ; Mahony *et al.* (2005)). Le filtre complémentaire permet de donner un poids entre la valeur fournie par l'accéléromètre et la valeur du gyroscope. Il peut être cependant difficile à avoir de bons résultats, le filtre étant trop simpliste. Le filtre de Kalman a été beaucoup utilisé pour ce genre de problème et est considéré comme une référence (Kalman *et al.* (1960) et Grover & Hwang (2012)). Une comparaison entre le filtre de Kalman et le filtre complémentaire a été faite par Higgins (1975). Le filtre de Kalman peut cependant être compliqué à réaliser lorsque l'on utilise des modèles non linéaires. Le filtre de Kalman étendue doit être utilisé (Marins *et al.* (2001)). Le temps d'exécution et la

complexité de l'implémentation de ce filtre peuvent devenir problématiques. D'autres solutions existent, notamment l'utilisation de système flou (Sung (2003)). Ce projet utilisera cependant le filtre utilisé par Madgwick *et al.* (2011). Ce filtre se base sur les quaternions afin d'éviter les problèmes liés aux angles d'Euler. Il utilise un algorithme simple et rapide d'exécution.

L'utilisation de ces algorithmes sur le gyroscope et l'accéléromètre fonctionne correctement lorsque les expériences se font sur un repère local. Les drones sont cependant beaucoup utilisés à l'extérieur avec comme coordonnées une position précise par rapport à la Terre. Il faut alors connaitre la direction du drone par rapport à celle-ci. On utilise pour cela le magnétomètre avec le gyroscope et l'accéléromètre. On parle alors d'un ensemble AHRS (Attitude and Heading Reference System). Les filtres évoqués précédemment peuvent être utilisés pour ajouter le magnétomètre.

1.5 La carte électronique

Afin d'implémenter le contrôleur, lire les différents capteurs et envoyer des signaux aux moteurs, une carte de commande doit être utilisée. Elle contient différents composants (microcontrôleur, ports entrées-sorties, module Ethernet, etc.). Il peut être fastidieux de réaliser une carte de contrôle à partir de zéro. De nombreuses compagnies ainsi que certaines organisations ont réalisé des cartes spécifiques à l'utilisation de drone. Une liste non exhaustive a été faite à la suite de ce sous-chapitre.

1.5.1 NAVIO

Les produits NAVIO (Figure 1.4) sont des cartes incluant un système d'exploitation embarqué basé sur Linux. Ils utilisent le nano-ordinateur Raspberry Pi permettant d'effectuer des opérations complexes grâce à la performance du processeur. Les produits NAVIO ont une carte WI-FI permettant la transmission rapide de données. Deux centres inertiels sont inclus permettant d'avoir un système de redondance. Un baromètre ainsi qu'un récepteur GPS (Global Positioning System) sont intégrés. Quatorze ports MLI sont intégrés (Emlid (2017b)).

Figure 1.4 Navio2
Tirée de Emlid (2017a)

1.5.2 Erle-Brain

La famille Erle-Brain (Figure 1.5) dispose d'un système d'exploitation graphique basé sur Linux contrairement aux NAVIO. Il dispose de quatre ports USB, d'un port audio, d'un câble HDMI, des deux ports I2C (Inter-Integrated Circuit) ainsi qu'un port UART (Universal Asynchronous Receiver Transmitter). Il a la particularité d'être directement compatible ROS (Robot Operating System). Une centrale inertielle à 9 degrés de liberté ainsi, un capteur de température et un baromètre sont inclus. Douze ports MLI sont disponibles ainsi qu'un port MPI (modulation en position d'impulsions) (Erle (2017a)).

Figure 1.5 Erle Brain v1.1
Tirée de Erle (2017b)

1.5.3 Pixhawk

Le Pixhawk (Figure 1.6) dispose d'un processeur d'une fréquence de 168 MHz avec une capacité d'effectuer 252 millions d'instructions à la seconde. Il dispose de plusieurs ports de périphéries(UART, I2C, CAN). Contrairement aux cartes évoquées précédemment, le Pixhawk ne dispose pas d'un système d'exploitation basé sur Linux, mais utilise Nuttx. Un système embarqué comme un Raspberry Pi ou un Odroid est souvent utilisé en parallèle du Pixhawk afin d'effectuer des opérations plus complexes. Il dispose d'une centrale inertielle ainsi que de 14 ports MLI. Cette carte combinée avec un Odroid XU4 sera utilisée pour ce projet.

Figure 1.6 Pixhawk
Tirée de Pixhawk (2017)

1.5.4 Autres

Les cartes vues précédemment peuvent être utilisées sur plusieurs types de véhicules différents (drones, rovers, bateaux). Ils existent cependant des drones où la carte, suivie des capteurs, est directement imposée. C'est le cas pour les micro drones ou encore du drone Parrot Bebop. Ce genre de pratique permet de garantir la comptabilité des différents périphériques utilisés pour le drone et d'avoir des performances optimales. Le Crazyflie, par exemple, dispose d'une carte électronique propre au drone ainsi que son micrologiciel. Ce micrologiciel, étant un code à source ouvert, permet le développement rapide et optimisé du drone. On utilisera le Crazyflie pour tester le contrôleur sur un micro drone.

1.6 Unité de gestion de vol

On a vu précédemment l'utilisation de cartes électroniques munies de microcontrôleur permettant d'implémenter le contrôleur du drone. On doit cependant programmer la carte correctement afin de pouvoir avoir des résultats optimaux. De nombreux paramètres doivent être pris en compte (gestion multi taches, gestion et calibrage des capteurs, protocole de communication). La programmation peut rapidement devenir une tâche laborieuse. Une solution à ce problème

est d'utiliser une unité de gestion de vol (*flight management unit* en anglais). L'unité de gestion de vol est un micrologiciel incluant toute la gestion des périphériques et des capteurs. Le protocole de communication entre l'ordinateur mère et le drone est éventuellement géré. Le micrologiciel est souvent prêt à l'emploi, l'utilisateur pouvant directement piloter un drone une fois le logiciel implémenté dans le microcontrôleur. Plusieurs de ces unités de gestion de vol sont un code à source ouvert. Il est alors facile de modifier le code afin de ne programmer que le contrôleur tout en évitant de programmer toutes les couches aux alentours (notamment le protocole de communication). Deux principales unités de systèmes de vol sont utilisées dans l'industrie du drone : PX4 et Ardupilot.

1.6.1 PX4

Px4 (disponible sur https://dev.px4.io/en/) est un micrologiciel fonctionnant principalement sur le système d'exploitation Nuttx. Il permet de contrôler plusieurs types d' engins tels que des voitures de course, des aéronefs à décollage et atterrissage verticaux, des drones à rotors multiples, ainsi que des drones à voilure fixe. Il permet d'utiliser le protocole de communication standard Mavlink (Micro Air Vehicle Link). Sa facilité de programmation permet d'ajouter facilement de nouvelles routines. Le code est complètement à source ouverte. PX4 dispose, par défaut, d'un contrôleur de position pouvant utiliser le GPS ou une information par capture de mouvement. Il dispose de plusieurs outils permettant la calibration des capteurs. Plusieurs cartes sont compatibles avec ce logiciel notamment les cartes évoquées précédemment. On décidera d'utiliser ce micrologiciel pour notre projet.

1.6.2 Ardupilot

Contrairement au PX4, Ardupilot peut-être directement installé sur le système d'exploitation Linux (voir http://ardupilot.org/dev/). Comme le PX4, le micrologiciel est un code à source ouvert. Il est aussi compatible avec le protocole de communication Mavlink. Le logiciel peut être utilisé sur des types d'engins comme PX4. Ardupilot est connu pour avoir une variété de contrôleurs permettant d'effectuer plusieurs types de missions (trajectoires acrobatiques,

maintien d'une position, atterrissage automatique ...). Il est compatible par défaut avec des GPS différentiels.

1.7 Technologies développées pour récupérer la position

La détection de la position du drone est primordiale à la conception du contrôleur. La technologie utilisée doit être la plus précise possible et doit avoir un taux de rafraichissement rapide. Différents capteurs sont utilisés suivant les exigences. Le baromètre ou le sonar peuvent être utilisés pour contrôler l'altitude du drone. Ils peuvent poser cependant des problèmes lorsque l'on veut contrôler l'aéronef dans toutes les directions. Le GPS est souvent utilisé pour avoir un contrôle complet du drone par rapport à la Terre (Koehl (2012)). Le GPS différentiel permet d'améliorer la précision du GPS. Il faut cependant disposer d'une référence terrestre pour utiliser ce dernier. Le GPS ne peut pas être utilisé en intérieur dû à l'impossibilité de capter un signal provenant d'un satellite. D'autres méthodes doivent être alors utilisées pour récupérer la position du drone dans un laboratoire. Deux méthodes principales existent : la vision par ordinateur et la capture de mouvement. L'utilisation de la technique de modulation radio ULB (Ultra Large Bande) commence à être perfectionnée pour avoir des résultats précis.

1.7.1 Vision par ordinateur

La vision par ordinateur est le traitement d'image effectué par un ordinateur à partir d'une ou plusieurs caméras. Les caméras sont généralement placées sur le drone. Des marqueurs, où leurs dimensions et positions sont connues, permettent à un ordinateur de calculer la position actuelle du drone à l'aide de différents algorithmes de traitement d'image (détection de contours, détection de couleur, etc.). Des chercheurs (Yang *et al.* (2013)) ont réussi à faire atterrir un micro drone avec une image de référence et en utilisant une caméra. Cette technique est cependant assez limitée et demande un algorithme complexe pour avoir la profondeur. La vision par ordinateur ne permet pas de faire des trajectoires complexes et précises. On préféra utiliser la capture des mouvements.

1.7.2 Capture des mouvements

La capture de mouvement est utilisée dans plusieurs domaines (audiovisuel, jeux vidéo, robotique...). Elle se compose généralement d'une ou plusieurs séries de caméra ou de capteurs spécifiques (centrales inertielles, capteur magnétique ou optique). Ces capteurs sont souvent suivis de marqueurs permettant la détection immédiate de l'objet à mesurer. Ces capteurs disposent d'une unité de calcul permettant de faire le traitement d'information afin de retrouver la position de l'objet. Trois types de capture sont souvent utilisés pour la détection de position pour un drone : l'Optitrack, le Vicon et la Kinect.

1.7.2.1 Optitrack

L'Optitrack est une série de caméras à infrarouge à taux de rafraichissement élevé (jusqu'à 340 Hertz). La précision de cette série de caméras peut être en dessous du millimètre (Chan *et al.* (2017)). Un marqueur réfléchissant permet à Optitrack de détecter directement l'objet. Trois marqueurs sur un objet peuvent alors détecter l'orientation. Les capteurs viennent directement avec leur propre algorithme de détection d'objet et sont donc prêts à l'emploi. Des expériences ont utilisé la vision par ordinateur ainsi qu'un système Optitrack afin de contrôler un drone suivant un être humain (Yao *et al.* (2017)). Dix caméras Optitrack ont été utilisées sur un drone afin de tester un contrôleur non linéaire par mode glissant (Robin (2017)).

1.7.2.2 Vicon

Le système Vicon est une série de caméras haute résolution permettant d'avoir la position et l'orientation d'un objet avec un taux de rafraichissement très élevé (pouvant aller jusqu'à 600 Hertz) ainsi qu'une très grande précision pouvant aller en dessous du millimètre (Windolf *et al.* (2008)). Ce système peut être utilisé à l'intérieur comme à l'extérieur sur une superficie allant de 20 mètres carrés. Plusieurs contrôleurs ont été implémentés avec succès. Le système Vicon permet d'implémenter facilement un contrôleur de position non linéaire (Subramanian (2015)).

Luis & Ny (2016) utilisent un modèle linéarisé avec le système Vicon afin de contrôler un micro drone.

1.7.2.3 Kinect

Les systèmes de capture de mouvement précédent peuvent s'avérer coûteux. Une solution bon marché est d'utiliser une caméra Kinect développée par Microsoft. La Kinect a été conçue pour le divertissement pour être utilisée à l'aide d'une console de jeux vidéo. Deux versions ont été développées : la Kinect v1 ainsi que la Kinect v2 (dit Kinect One). Sarbolandi *et al.* (2015) montrent les performances entre les différentes versions. La Kinect v2 utilise une résolution haute définition et deux caméras infrarouges afin de détecter la position d'un objet. Une caméra couleur est aussi disponible ainsi qu'un microphone pour la reconnaissance vocale. Le taux de rafraichissement est cependant faible par rapport aux systèmes précédents (30 Hertz). On peut avoir une précision à l'échelle du millimètre sur une distance maximale d'environ quatre mètres. On utilisera ce système pour notre projet.

1.8 Commandes utilisées pour contrôler un quadcopter

Afin de contrôler un quadcopter en position et en orientation, un système de commande doit être mis en place. Plusieurs types de commandes peuvent être utilisés. Une liste non exhaustive est présentée dans ce chapitre.

1.8.1 Commandes linéaires

Les unités de gestion de vol utilisent en général des contrôleurs linéaires. La commande la plus utilisée est le PID (Proportionnel Intégrateur Dérivé). Cette commande a été utilisée plusieurs fois (Hoffmann *et al.* (2007) ; Yang *et al.* (2010)). Les termes d'intégration et de la dérivation permettent d'obtenir une stabilité ainsi qu'une convergence de l'erreur à zéro de manière exponentielle. Ce contrôleur n'a pas besoin de connaître le modèle dynamique du quadcopter. Une amélioration à ce contrôleur est la commande linéaire quadratique (LQR) qui permet de faire

un suivi de trajectoire en optimisant la consommation d'énergie. Jafari *et al.* (2010) utilisent ce contrôleur avec succès avec un temps de réponse lent. Ces commandes permettent d'être implémentées facilement et ont le bénéfice de pouvoir être utilisées sur différents types de drone sans connaitre la dynamique. Il peut être cependant difficile d'optimiser les gains facilement afin d'avoir des temps de réponses optimales. Il peut aussi être difficile d'utiliser ces contrôleurs pour faire des trajectoires plus compliquées (trajectoires acrobatiques par exemple).

1.8.2 Commande non linéaire par backstepping

Cette commande décrite par Kanellakopoulos *et al.* (1991) a permis d'utiliser la dynamique du drone ainsi que la théorie de Lyapunov afin d'assurer la stabilité du contrôleur. Le principe consiste à diviser le système en plusieurs sous-systèmes en cascade. Les lois de commandes sont alors faites pour chaque sous système, de manière décroissante, jusqu'à avoir une loi de commande globale pour tout le système. Plusieurs recherches sur le quadcopter ont été faites afin d'utiliser ce contrôleur. Das *et al.* (2009) et Raffo *et al.* (2008) utilisent le contrôleur backstepping en position et en attitude. Il se peut cependant que l'incertitude des paramètres puisse engendrer des erreurs de position. Amiri *et al.* (2013) utilisent alors une action intégrale pour converger les erreurs à zéro. L'action intégrale entraine cependant un dépassement au début de la trajectoire. Des commandes backstepping adaptatives ont été mises en place afin d'estimer les paramètres de la dynamique du quadcopter (Zuo (2013)). Ces commandes adaptatives permettent aussi d'estimer des forces extérieures permettant d'assurer une convergence de l'erreur à zéro.

1.8.3 Commande par mode glissant

La commande par mode glissant permet, à partir de la théorie de Lyapunov, d'obtenir une surface de glissement où les variables d'états du système à contrôler sont maintenues dans cette région. Plusieurs recherches telles que Bouadi & Tadjine (2007) et Benallegue *et al.* (2008) démontrent la stabilité de la commande avec des paramètres dynamiques et perturbations extérieures inconnues. Cette commande dispose d'un problème bien particulier appelé *chattering*.

Ce phénomène peut provoquer des secousses au système. Plusieurs techniques sont utilisées pour essayer de réduire ce phénomène. Bandyopadhyay *et al.* (2013) utilisent un contrôleur permettant de modifier les gains de manière dynamique afin de réduire le phénomène. González *et al.* (2014) utilisent une fonction de saturation afin de corriger le problème. Des fonctions linéaires, comme une fonction avec une loi exponentielle, permettent (Fallaha *et al.* (2011)) de réduire le *chattering*. Lee *et al.* (2009) utilisent une commande adaptative avec la commande par mode glissant afin de contrôler un quadcopter et comparent cela à un contrôleur de linéarisation entrée-sortie.

1.8.4 Linéarisation entrée-sortie

La linéarisation entrée-sortie permet d'utiliser des lois de commande linéaires sur des systèmes non linéaires. Le principe est d'utiliser un changement de variable afin de trouver une variable d'état linéaire décrivant la dynamique du drone. Yaou *et al.* (2013) proposent une linéarisation entrée-sortie en utilisant les crochets de Lie pour contrôler les quadcopters. Il est possible, à partir de la théorie de Lypunov, d'utiliser une commande adaptative afin de contrôler un drone ayant des paramètres inconnus ainsi que des perturbations extérieures. Une fois la variable d'état linéarisée, il devient simple d'implémenter n'importe quelles lois linéaires. Cette méthode, étant facile à réaliser et facilement modifiable, sera implémentée pour ce projet.

1.9 Conclusion

Ce chapitre a énuméré les différentes technologies utilisées pour contrôler un quadcopter. On a vu plusieurs moyens d'implémentation permettant de faciliter la réalisation du contrôleur. Plusieurs types de capteurs peuvent être utilisés afin de récupérer la position et l'orientation du drone. On utilisera pour ce projet le Pixawk avec le micrologiciel PX4 pour implémenter le contrôleur sur un drone. Le micro drone Crazyflie sera aussi utilisé pour tester l'efficacité du contrôleur sur ce genre d'engin. Une linéarisation entrée-sortie sera faite pour commander le drone.

CHAPITRE 2

MODÉLISATION DU QUADCOPTER

2.1 Fonctionnement théorique du drone

Le quadcopter est constitué de 4 moteurs permettant d'orienter le drone (Figure 2.1). Les vitesses ω_1 et ω_3 des moteurs M_1 et M_3 vont dans le sens inverse des aiguilles d'une montre et les vitesses ω_2 et ω_4 des moteurs M_2 et M_4 dans le sens des aiguilles d'une montre. Le quadcopter peut s'orienter de différentes manières : la configuration dite en plus, dans laquelle le nez du drone se situe en face du moteur M_1 et la configuration en croix (ou *cross* en anglais) où le nez se trouve entre les moteurs M_1 et M_2. Une rotation positive de l'angle en lacet s'effectue en augmentant les vitesses des moteurs M_2 et M_4 par rapport aux vitesses des moteurs M_1 et M_3. Une augmentation des vitesses des moteurs M_1 et M_3 par rapport aux vitesses des moteurs M_2 et M_4 produira une rotation en lacet négative.

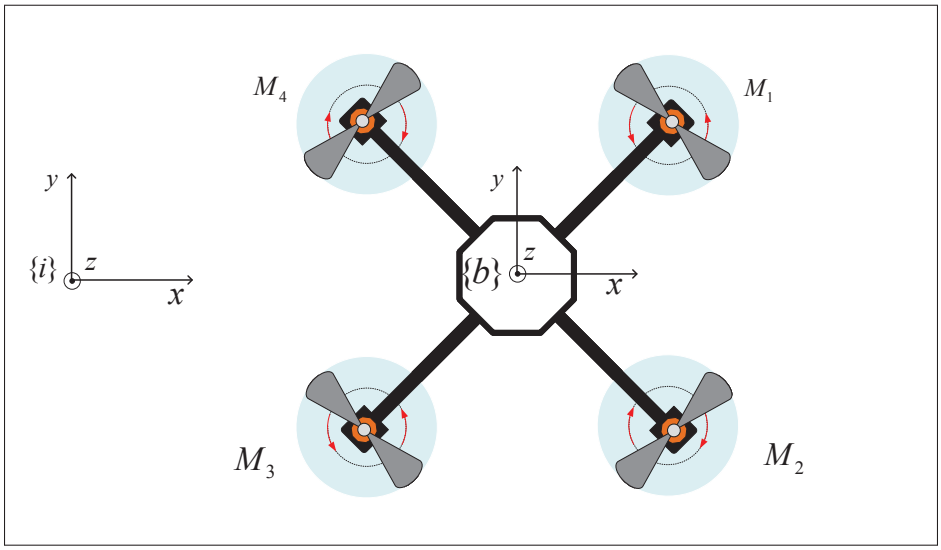

Figure 2.1 Repères principaux du drone

Lorsque le drone est dans une configuration en croix, la différence de vitesses des quatre moteurs est utilisée pour effectuer une rotation en roulis ou en tangage. En effet, une augmentation

des vitesses des moteurs M_1 et M_4 par rapport aux vitesses des moteurs M_2 et M_3 permet d'avoir une rotation en tangage positive et inversement pour une rotation négative. Une augmentation des vitesses des moteurs M_1 et M_2 par rapport aux vitesses des moteurs M_3 et M_4 réalise une rotation positive en roulis et inversement pour une rotation négative.

Lorsque le drone est dans une configuration en plus, la différence de vitesses de deux moteurs est utilisée pour faire les rotations en roulis et en tangage. Une augmentation de vitesse du moteur M_1 par rapport à M_3 fera une rotation positive en tangage. Une augmentation de la vitesse du moteur M_4 par rapport à M_2 fera une rotation positive en roulis.

Deux repères principaux sont utilisés pour décrire le mouvement du drone. Le repère $\{i\}$ représente le repère inertiel. Il est fixe par rapport à la Terre. Le repère $\{b\}$ représente le repère du corps du drone. L'axe z du le repère $\{b\}$ est toujours normal au corps du drone. Le repère $\{b\}$ a été pris de sorte à être dans une configuration en croix.

L'orientation du drone peut être représentée de manières différentes. On peut, en effet, utiliser les quaternions (Jia (2016)) ou les angles d'Euler. Contrairement aux angles d'Euler, les quaternions n'ont pas besoin d'avoir des repères auxiliaires pour être décrits proprement. Ils peuvent aussi éviter le phénomène, dit blocage de cadran (ou *gimbal lock* en anglais), qui enlève deux degrés de liberté au drone. Ce phénomène ainsi que les quaternions seront expliqués plus en détail dans la suite de ce chapitre.

2.2 Changement de repère entre le repère inertiel et le repère du corps du drone

Trois repères auxiliaires $\{r_i\}$,$\{r_\psi\}$ et $\{r_\theta\}$ doivent être utilisés pour décrire les angles d'Euler. On utilisera la convention lacet-tangage-roulis à savoir (Figure 2.2) :

a. L'origine du repère $\{r_i\}$ se trouve au centre du corps du drone. Son orientation est la même que l'orientation du repère inertiel $\{i\}$;

b. Le repère $\{r_\psi\}$ suit une rotation d'angle ψ sur l'axe z du repère $\{i\}$;

c. Le repère $\{r_\theta\}$ suit une rotation d'angle θ sur l'axe y du repère r_ψ ;

d. Le repère $\{r_\theta\}$ suit une rotation d'angle ϕ sur l'axe x du repère r_θ et est confondu avec le repère $\{b\}$.

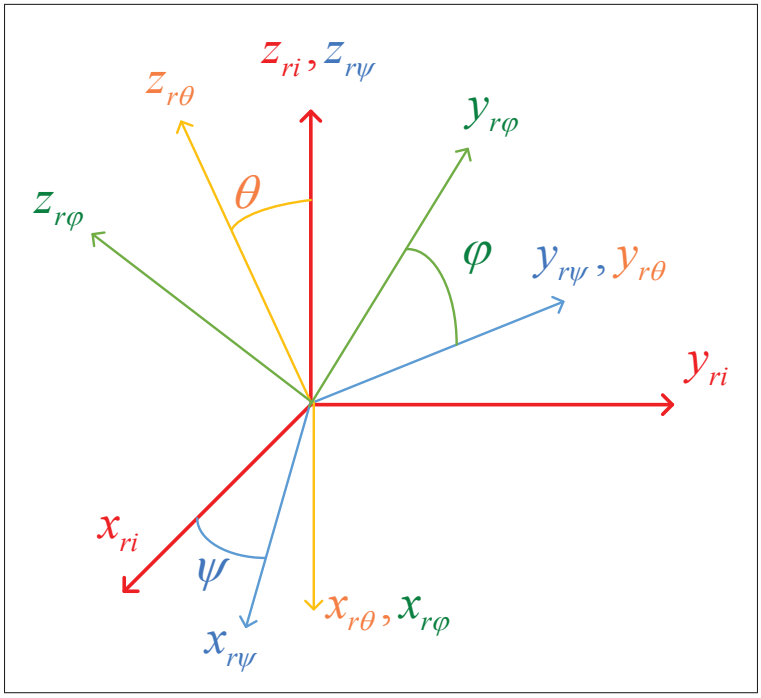

Figure 2.2 Repères auxiliaires du drone

Des matrices de rotation permettent de décrire les changements de repère. On appellera ${}^{x}_{y}R$ la matrice de rotation permettant de passer du repère $\{y\}$ au repère $\{x\}$. On a alors les matrices de rotation suivantes :

$$ {}^{r_\theta}_{r_\psi}\boldsymbol{R} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.1} $$

$$ {}^{b}_{r_\theta}\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2.2} $$

$$ {}^{r_\psi}_{r_i}\boldsymbol{R} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3} $$

On peut, à partir des équations (2.3), (2.1) et (2.2), retrouver la matrice de rotation ${}^{b}_{r_i}R$ permettant de passer du repère inertiel $\{r_i\}$ au repère du corps $\{b\}$:

$$ {}^{b}_{r_i}\boldsymbol{R} = {}^{b}_{r_\theta}\boldsymbol{R}\,{}^{r_\theta}_{r_\psi}\boldsymbol{R}\,{}^{r_\psi}_{r_i}\boldsymbol{R} \tag{2.4} $$

$$ {}^{b}_{r_i}\boldsymbol{R} = \begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ \sin(\phi)\sin(\theta)\cos(\psi)-\cos(\phi)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi)+\cos(\phi)\cos(\psi) & \sin(\phi)\cos(\theta) \\ \cos(\phi)\sin(\theta)\cos(\psi)+\sin(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\sin(\psi)-\sin(\phi)\cos(\psi) & \cos(\phi)\cos(\theta) \end{bmatrix} \tag{2.5} $$

Autrement dit, un vecteur ${}^{i}X$ exprimé dans le repère $\{i\}$ peut être exprimé dans le repère $\{b\}$ par un vecteur ${}^{b}X$ à l'aide de l'expression suivante :

$$ {}^{b}\boldsymbol{X} = {}^{b}_{i}\boldsymbol{R}\,{}^{i}\boldsymbol{X} = {}^{b}_{r_i}\boldsymbol{R}\,{}^{i}\boldsymbol{X} \tag{2.6} $$

2.3 Vitesses angulaires

Le quadcopter dispose d'un gyroscope permettant de relever les vitesses angulaires du drone par rapport au repère $\{i\}$ et exprimées dans le repère $\{b\}$. Ces vitesses peuvent être symbolisées par un vecteur ${}^{b}\boldsymbol{\omega}$. Ce vecteur est composé de trois coordonnées p,q,r décrivant respectivement les vitesses angulaires autour des axes x, y et z du repère $\{b\}$:

$$ {}^{b}\boldsymbol{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.7} $$

Afin de pouvoir implémenter une commande permettant de contrôler l'orientation du drone, les vitesses des angles d'Euler doivent être exprimées. Il est donc important de trouver une relation

entre les données venant du gyroscope et les angles. Il peut être démontré (Craig (2005)) que le changement de repère d'une vitesse angulaire d'un repère à un autre est :

$$^{i+1}\boldsymbol{\omega}_{i+1} = {}^{i+1}_{i}\boldsymbol{R}\,{}^{i}\boldsymbol{\omega}_{i} + \dot{\boldsymbol{\theta}}_{i+1}\,{}^{i+1}\hat{\boldsymbol{Z}}_{i+1} \tag{2.8}$$

Avec $^{i+1}\omega_{i+1}$ la vitesse angulaire du repère $\{i+1\}$ par rapport au repère inertiel exprimé dans le repère$\{i+1\}$, ${}^{i+1}_{i}R^{i}$ la matrice de rotation passant de l'ancien repère au nouveau, $\dot{\theta}_{i+1}$ la vitesse angulaire de l'angle de rotation et $^{i+1}\hat{Z}_{i+1}$ le vecteur unitaire colinéaire avec l'axe de rotation dans le nouveau repère. On peut, à partir de (2.8) et par récurrence, en déduire :

$$^{b}\boldsymbol{\omega} = \begin{bmatrix}\dot{\phi}\\0\\0\end{bmatrix} + {}^{b}_{r_\theta}\boldsymbol{R}\begin{bmatrix}0\\\dot{\theta}\\0\end{bmatrix} + {}^{b}_{r_\theta}\boldsymbol{R}^{r_\theta}_{r_\psi}\boldsymbol{R}\begin{bmatrix}0\\0\\\dot{\psi}\end{bmatrix} \tag{2.9}$$

$$^{b}\boldsymbol{\omega} = \begin{bmatrix}1 & 0 & -\sin(\theta)\\0 & \cos(\phi) & \sin(\phi)\cos(\theta)\\0 & -\sin(\phi) & \cos(\phi)\cos(\theta)\end{bmatrix}\begin{bmatrix}\dot{\phi}\\\dot{\theta}\\\dot{\psi}\end{bmatrix} \tag{2.10}$$

$$^{b}\boldsymbol{\omega} = \boldsymbol{J}\dot{\boldsymbol{\eta}} \tag{2.11}$$

2.4 Relation entre les angles d'Euler et les quaternions

Afin de voir les limites de la modélisation avec les angles d'Euler, la matrice J doit être inversée. On se retrouve avec ce résultat :

$$\boldsymbol{J}^{-1} = \begin{bmatrix}1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta)\\0 & \cos(\phi) & -\sin(\phi)\\0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta)\end{bmatrix} \tag{2.12}$$

La relation (2.12) permet de voir les limites de la modélisation du drone avec les angles d'Euler. $\ddot{\phi}$ et $\ddot{\psi}$ sont indéfinis lorsque θ est à 90 degrés. Le drone perd alors deux degrés de liberté. On appelle ce phénomène blocage de cadran. Utiliser les angles d'Euler interdit au drone de faire des trajectoires plus compliquées (faire des trajectoires acrobatiques par exemples). Ce projet se limitant à faire des trajectoires simples, l'orientation du drone n'atteindra jamais ces singularités.

Ces singularités peuvent cependant poser un problème lorsque l'on essaie de récupérer les angles d'orientation à l'aide de capteurs (gyroscope, accéléromètre et magnétomètre). Les drones utilisés dans ce projet utilisent les quaternions pour relever les orientations, il est donc important de comprendre les fondamentaux ainsi que la relation entre les angles d'Euler et les quaternions. Les quaternions permettent de décrire l'orientation d'un repère en utilisant un vecteur unitaire et un angle θ_{ref} (voir Figure 2.3).

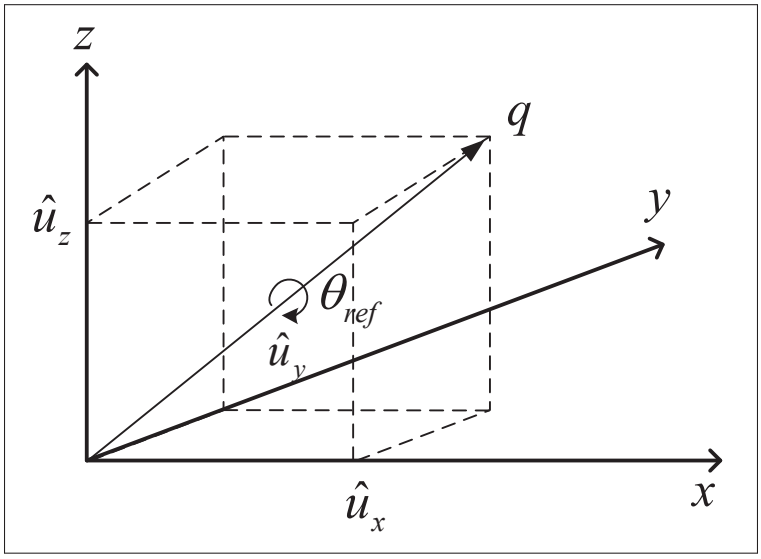

Figure 2.3 Représentation d'un quaternion

Une manière de définir un quaternion q est de définir ses éléments de la manière suivante :

$$\boldsymbol{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix} = \begin{bmatrix} \cos(\frac{\theta_{ref}}{2}) & \hat{u}_x \sin(\frac{\theta_{ref}}{2}) & \hat{u}_y \sin(\frac{\theta_{ref}}{2}) & \hat{u}_z \sin(\frac{\theta_{ref}}{2}) \end{bmatrix} \tag{2.13}$$

Avec $\hat{u}_x$, $\hat{u}_y$ et $\hat{u}_y$ les éléments du vecteur unitaire. Le conjugué du quaternion q est représenté alors de la manière suivante :

$$\boldsymbol{q}_{conj} = \begin{bmatrix} q_1 & -q_2 & -q_3 & -q_4 \end{bmatrix} \tag{2.14}$$

Si l'on définit ${}^a_b q$ le quaternion permettant de décrire l'orientation d'un repère $\{b\}$ par rapport à un repère $\{a\}$ et v_a un vecteur donné dans le repère $\{a\}$, la relation décrivant le vecteur v_a dans le repère $\{b\}$ est :

$$\begin{bmatrix} 0 & {}^b\boldsymbol{v} \end{bmatrix} = {}^b_a\boldsymbol{q} \otimes \begin{bmatrix} 0 & {}^a\boldsymbol{v} \end{bmatrix} \otimes {}^b_a\boldsymbol{q}_{conj} \tag{2.15}$$

Le symbole $\otimes$ étant le produit des quaternions en utilisant le principe de Hamilton (Jia (2016)). On peut alors, à partir des relations (2.13), (2.14) et (2.15), établir la matrice de rotation de l'orientation du drone en fonction des éléments du quaternion soit :

$${}^i_b\boldsymbol{R} = \begin{bmatrix} 2q_1^2 - 1 + 2q_2^2 & 2(q_2q_3 - q_1q_4) & 2(q_2q_4 + q_1q_3) \\ 2(q_2q_3 + q_1q_4) & 2q_1^2 - 1 + 2q_3^2 & 2(q_3q_4 - q_1q_2) \\ 2(q_2q_4 - q_1q_3) & 2(q_3q_4 + q_1q_2) & 2q_1^2 - 1 + 2q_4^2 \end{bmatrix} \tag{2.16}$$

À partir de la matrice de rotation inverse de l'équation (2.5) et par identification, on se retrouve avec les expressions suivantes :

$$\psi = \arctan 2(2q_2q_3 + 2q_1q_4, 2q_1^2 - 1 + 2q_2^2) \tag{2.17}$$

$$\theta = -\arcsin(2q_2q_4 - 2q_1q_3) \tag{2.18}$$

$$\phi = \arctan 2(2q_3q_4 + 2q_1q_2, 2q_1^2 - 1 + 2q_4^2) \tag{2.19}$$

Comme dit précédemment, l'utilisation des quaternions permet d'éviter des singularités. On remarque cependant dans l'équation (2.18) une possibilité d'avoir plusieurs solutions.

Ce phénomène peut se retrouver lors de la conception du contrôleur de position, une étude de cas doit alors être faite.

2.5 Dynamique du drone

Les forces exercées sur le drone peuvent être exprimées dans le repère inertiel avec les lois de Newton :

$$m\ddot{\boldsymbol{X}}_i = {}^i_b\boldsymbol{R}\,{}^b\boldsymbol{F} - mg\begin{bmatrix}0\\0\\1\end{bmatrix} - m\boldsymbol{\Delta} \tag{2.20}$$

Avec m la masse du drone en kg et $\ddot{X}_i$ les accélérations du drone en m.s^{-2} dans le repère inertiel soit :

$$\ddot{\boldsymbol{X}}_i = \begin{bmatrix}\ddot{x}_i\\\ddot{y}_i\\\ddot{z}_i\end{bmatrix} \tag{2.21}$$

Le vecteur ${}^bF$ représente la force (*thrust* en anglais) de portance exprimée dans le repère $\{b\}$. Cette force ne prend effet que sur l'axe z du repère $\{b\}$. On a donc :

$${}^b\boldsymbol{F} = \begin{bmatrix}0\\0\\T\end{bmatrix} \tag{2.22}$$

Avec T la valeur de la force de portance en N.

$m\Delta$ représente des forces extérieures s'exerçant sur les drones. Ces forces peuvent provenir de plusieurs sources (vents, charges...). Elles ne sont pas connues. Δ est un vecteur de la forme :

$$\boldsymbol{\Delta} = \begin{bmatrix} \Delta_x \\ \Delta_y \\ \Delta_z \end{bmatrix} \tag{2.23}$$

On peut noter que les forces dues à la résistance de l'air ne sont pas incluses. Le drone volera en faible vitesse créant ainsi des forces de la résistance de l'air faible par rapport aux autres forces. Les relations de Newton ainsi que les effets de Coriolis stipulent que :

$${}^{b}\boldsymbol{F}_{totale} = m({}^{b}\dot{v} + {}^{b}\boldsymbol{\omega} \times {}^{b}v) \tag{2.24}$$

Avec ${}^{b}F_{totale}$ les forces mentionnées dans l'équation (2.20) exprimées dans le repère $\{b\}$ et ${}^{b}v$ un vecteur des vitesses linéaires du drone par rapport au repère inertiel exprimées dans le repère {b} dont ses éléments sont :

$${}^{b}\boldsymbol{v} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.25}$$

En modifiant l'égalité dans l'équation (2.24) et en utilisant les vecteurs (2.7) et (2.25) on obtient :

$$\begin{bmatrix} \dot{u} \\ v \\ w \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{{}^{b}\boldsymbol{F}_{totale}}{m} \tag{2.26}$$

Les équations du moment d'Euler indiquent que :

$$\boldsymbol{u}_{\Theta} = \boldsymbol{I}^{b}\dot{\boldsymbol{\omega}} + {}^{b}\boldsymbol{\omega} \times (\boldsymbol{I}^{b}\boldsymbol{\omega}) \tag{2.27}$$

Avec :

$$\boldsymbol{u}_{\Theta} = \begin{bmatrix} u_{\phi} \\ u_{\theta} \\ u_{\psi} \end{bmatrix} \tag{2.28}$$

Et I le tenseur d'inertie. On peut ajouter dans l'expression (2.27) les couples créés par la rotation des hélices dues à un effet gyroscopique. Ces couples sont exprimés par la relation :

$$\boldsymbol{\tau}_g = \boldsymbol{\omega} \times \begin{bmatrix} 0 \\ 0 \\ J_m \Omega_r \end{bmatrix} \tag{2.29}$$

Avec J_m l'inertie d'un des rotors des moteurs et Ω_r la vitesse angulaire relative des hélices soient :

$$\Omega_r = \omega_1 + \omega_3 - \omega_2 - \omega_4 \tag{2.30}$$

Les trajectoires utilisées dans ce projet sont lentes. Les angles de tangage et de roulis sont au voisinage de zéro tout au long des expérimentations. Les effets gyroscopiques, dans ces conditions, influencent peu le drone. Ils peuvent donc être négligés. On néglige aussi les couples créés dus à la résistance à l'air pour les mêmes raisons évoquées précédemment pour les forces.

Le drone peut être considéré comme étant symétrique par rapport au repère $\{b\}$. L'origine du repère $\{b\}$ est située au centre du drone. Le drone est considéré comme un corps rigide. On peut alors considérer le tenseur d'inertie comme étant une matrice diagonale de la forme :

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.31}$$

On décrit, avec les équations (2.11), (2.20) et (2.27), la dynamique du drone en un vecteur d'état de la forme :

$$\dot{\boldsymbol{X}} = \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}) \tag{2.32}$$

Soit :

$$\begin{bmatrix} \ddot{x}_i \\ \dot{x}_i \\ \ddot{y}_i \\ \dot{y}_i \\ \ddot{z}_i \\ \dot{z}_i \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} {}^iF_x/m - \Delta_x \\ \dot{x}_i \\ {}^iF_y/m - \Delta_y \\ \dot{y}_i \\ {}^iF_z/m - g - \Delta_z \\ \dot{z}_i \\ p + q\sin(\phi)\tan(\theta) + r\cos(\phi)\tan(\theta) \\ q\cos(\phi) - r\sin(\phi) \\ q\sin(\phi)\sec(\theta) + r\cos(\phi)\sec(\theta) \\ (I_{yy} - I_{zz})qr/I_{xx} + u_\phi/I_{xx} \\ (I_{zz} - I_{xx})pr/I_{yy} + u_\theta/I_{yy} \\ (I_{xx} - I_{yy})pq/I_{zz} + u_\psi/I_{zz} \end{bmatrix} \tag{2.33}$$

Avec :

$${}^i\boldsymbol{F} = \begin{bmatrix} {}^iF_x \\ {}^iF_y \\ {}^iF_z \end{bmatrix} = {}^i_b\boldsymbol{R}\,{}^b\boldsymbol{F} \tag{2.34}$$

$$\boldsymbol{U} = \begin{bmatrix} {}^i\boldsymbol{F} \\ \boldsymbol{u}_\Theta \end{bmatrix} \tag{2.35}$$

2.6 Relation entre les forces et les couples du drone et des moteurs

Les forces et les couples vus dans la dynamique du drone vont être utilisés afin de concevoir le contrôleur du drone. On commande cependant les 4 moteurs du drone décrit dans le fonctionnement théorique du drone. Il est important de trouver une relation entre les forces et les couples des moteurs et du drone. Soit F_n et τ_n la force et le couple produits par le moteur M_n. En prenant l la longueur entre l'axe d'un moteur et le centre du drone, on retrouve les relations

suivantes dans une configuration en croix :

$$u_\phi = \frac{l(F_3 + F_4 - F_1 - F_2)}{\sqrt{2}} \tag{2.36}$$

$$u_\theta = \frac{l(F_1 + F_4 - F_2 - F_3)}{\sqrt{2}} \tag{2.37}$$

$$u_\psi = \tau_2 + \tau_4 - \tau_1 - \tau_3 \tag{2.38}$$

$$T = F_1 + F_2 + F_3 + F4 \tag{2.39}$$

Les vitesses des moteurs, leurs forces et leurs couples sont reliés par les relations (Deters *et al.* (2014)) :

$$F_n = \rho D^4 C_T(\omega_n)\omega_n^2 \tag{2.40}$$

$$\tau_n = \frac{\rho D^5}{2\pi} C_P(\omega_n)\omega_n^2 \tag{2.41}$$

Avec n le numéro du moteur d'une hélice, ρ la densité de l'air et D le diamètre d'une hélice. C_T et C_P sont des fonctions dépendantes des vitesses des moteurs. Elles dépendent également des caractéristiques des hélices. Ces fonctions sont en pratiques approximées par des constances déduites expérimentalement. On utilisera un appareil de mesures permettant de mesurer la force et le couple produit par un moteur afin d'interpoler ces fonctions.

L'appareil de mesure permettra aussi de mesurer la force du moteur en fonction de la tension appliquée à celui-ci. Il est donc utile d'inverser les relations (2.36), (2.37), (2.38) et (2.39). On peut déduire, à partir de (2.40) et (2.41), la relation :

$$\tau_n = \frac{D}{2\pi}\frac{C_P}{C_T}F_n = f(\omega_n)F_n \tag{2.42}$$

On a alors :

$$\begin{bmatrix} T \\ u_\phi \\ u_\theta \\ u_\psi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\frac{l}{\sqrt{2}} & -\frac{l}{\sqrt{2}} & \frac{l}{\sqrt{2}} & \frac{l}{\sqrt{2}} \\ \frac{l}{\sqrt{2}} & -\frac{l}{\sqrt{2}} & -\frac{l}{\sqrt{2}} & \frac{l}{\sqrt{2}} \\ -f(\omega_n) & f(\omega_n) & -f(\omega_n) & f(\omega_n) \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{2.43}$$

On en conclut :

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & -\frac{\sqrt{2}}{4l} & \frac{\sqrt{2}}{4l} & -\frac{1}{4f(\omega_n)} \\ \frac{1}{4} & -\frac{\sqrt{2}}{4l} & -\frac{\sqrt{2}}{4l} & \frac{1}{4f(\omega_n)} \\ \frac{1}{4} & \frac{\sqrt{2}}{4l} & -\frac{\sqrt{2}}{4l} & -\frac{1}{4f(\omega_n)} \\ \frac{1}{4} & \frac{\sqrt{2}}{4l} & \frac{\sqrt{2}}{4l} & \frac{1}{4f(\omega_n)} \end{bmatrix} \begin{bmatrix} T \\ u_\phi \\ u_\theta \\ u_\psi \end{bmatrix} \tag{2.44}$$

2.7 Conclusion

On a vu les différentes relations permettant de modéliser le drone. Ceci nous a permis de récupérer les équations décrivant les couples et les forces du drone. On a enfin établi la relation entre les forces, les couples du drone et les forces des moteurs. On doit à présent trouver les paramètres du drone afin de concevoir la commande.

CHAPITRE 3

MESURE DES PARAMÈTRES

Le chapitre précédent présente une modélisation de la dynamique du drone où les moments d'inertie ainsi que la masse du drone doivent être connus. La masse d'un drone peut être pesée facilement. Les moments d'inertie sont cependant plus compliqués à obtenir. Un appareil de mesure doit aussi être utilisé pour déterminer la relation entre les vitesses des hélices et les forces exercées. On essayera dans ce chapitre de trouver les paramètres du drone S500. On abordera une méthode théorique, expérimentale et une basée sur un modèle de simulation.

3.1 Drone basé sur le châssis S500

Le drone utilisé sera un quadcopter où le châssis S500 sera pris comme base de la structure (Figure 3.1). Il est muni de plusieurs éléments :

a. La carte électronique est le Pixhawk muni du micrologiciel PX4. Il sera utilisé pour implémenter le contrôleur et permettra de commander les moteurs ;

b. Le micro-ordinateur Odroid XU4 muni d'un dongle WI-FI permettra de faire une relation en WI-FI entre l'ordinateur, où les trajectoires et les informations de la Kinect seront envoyées, et le Pixhawk ;

c. Une batterie Turnigy 3300mAh permettant d'avoir une autonomie d'environ dix minutes ;

d. On utilisera une télécommande Spektrum DX6i suivie de son receveur de canaux AR610 ;

e. Quatre moteurs sans balais Multistar Elite 2216 KV920 munis de ses contrôleurs de vitesse, Multistar SBEC4A Turnigy 20A.

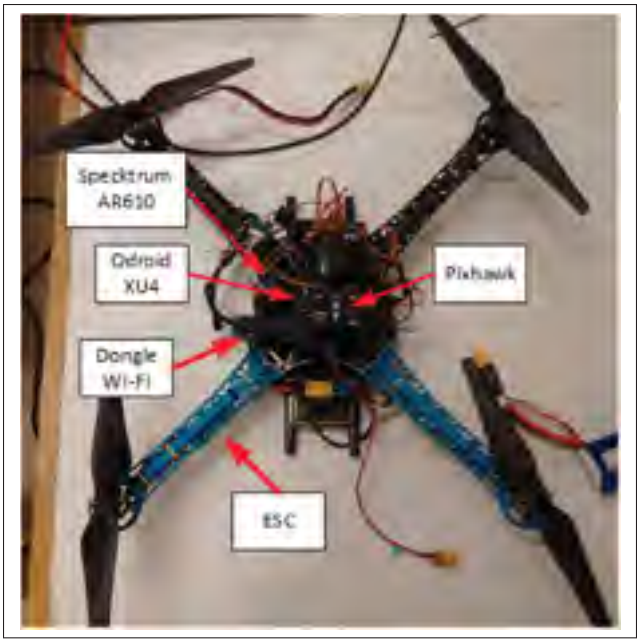

Figure 3.1 Drone S500

3.2 Mesure des moments d'inertie

3.2.1 Méthode théorique

On peut considérer le quadcotper comme étant un corps rigide où la majorité de la masse m_{base} est répartie dans un cylindre de rayon r_{base} et de hauteur h_{base} situé au centre du drone. Les bras sont considérés comme des tiges d'une longueur l_{bras} et de masse m_{bras}. Les moteurs sont modélisés par des cylindres de rayon r_{moteur}, de hauteur h_{moteur} et de masse m_{moteur} (les masses des hélices sont incluses). La Figure 3.2 montre la modélisation.

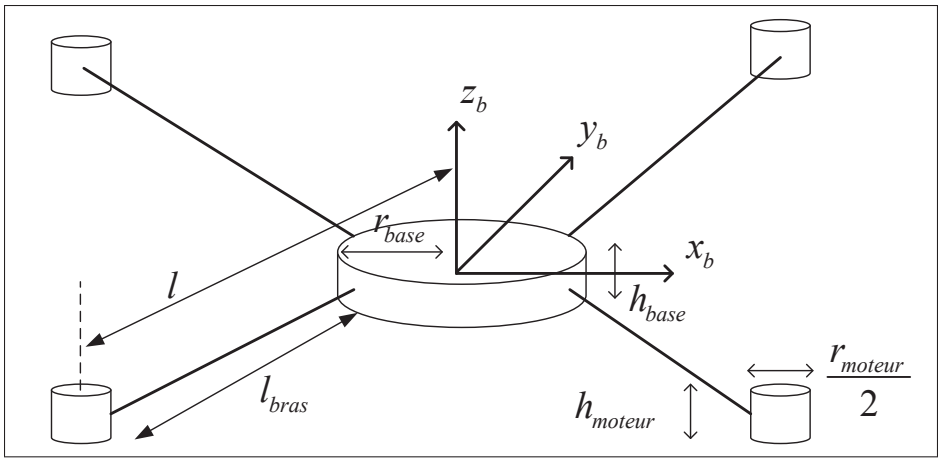

Figure 3.2 Modèle théorique d'un quadcopter

On peut alors, à partir du théorème des axes parallèles ainsi que les formules des moments d'inertie d'une tige et d'un cylindre, en déduire :

$$I_{zz} = 4\left(\frac{m_{moteur}r_{moteur}^2}{2} + m_{moteur}l^2\right) + \frac{m_{base}r_{base}^2}{2} + \frac{m_{bras}l_{bras}^2}{3} + m_{bras}l_{bras}^2 \tag{3.1}$$

$$I_{xx} = 4\left(\frac{m_{moteur}(3r_{moteur}^2 + h_{moteur}^2)}{12} + \frac{m_{moteur}l^2}{2}\right) + \frac{m_{base}(3r_{base}^2 + h_{base}^2)}{12} + \frac{m_{bras}l_{bras}^2}{6} + \frac{m_{bras}l_{bras}^2}{2} \tag{3.2}$$

$$I_{yy} = I_{xx} \tag{3.3}$$

On utilisera cette méthode pour le drone S500 afin de comparer les résultats entre la théorie et l'expérimentation. Le tableau 3.1 montre les caractéristiques du drone S500.

Tableau 3.1 Propriétés du drone S500

m_{bras} (kg)	m_{base} (kg)	r_{base} (m)	l_{bras} (m)	m_{moteur} (kg)	r_{moteur} (m)	h_{base} (m)	l (m)	h_{moteur} (m)
0,0704	0,7662	0,08	0,18	0,0766	0,014	0,085	0,24	0,031

On retrouve alors les résultats :

$$I_{xx} = 0{,}0122 \text{ kg} \cdot \text{m}^2 \tag{3.4}$$

$$I_{yy} = I_{xx} \tag{3.5}$$

$$I_{zz} = 0{,}0232 \text{ kg} \cdot \text{m}^2 \tag{3.6}$$

3.2.2 Méthode expérimentale

La méthode précédente peut être une bonne approximation si les masses des différents composants sont réparties de manière équilibrée. Une solution alternative est d'utiliser une manière expérimentale. Une des manières possibles est la méthode du pendule tri filaire. La méthode consiste à utiliser un disque d'une masse m_{disque} et dont la distance entre le centre du disque et les fils est de r_{disque}. Les trois fils suspendus au plafond ont une longueur l_{fils}. Les fils sont accrochés au disque de sorte à être séparés de 120 degrés par rapport au centre. On fera une rotation du disque sur l'axe c (voir Figure 3.3).

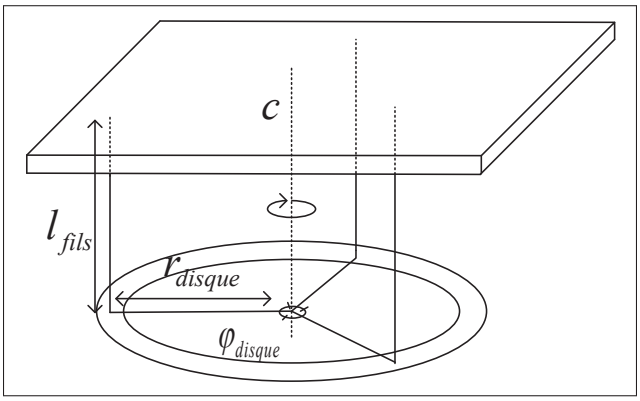

Figure 3.3 Méthode du pendule tri filaire

On peut alors démontrer (Harris & Piersol (2002)) que le moment d'inertie I_{disque} sur l'axe c lorsqu'une oscillation sur cet axe se produit est de :

$$I_{disque} = \frac{m_{disque} g r_{disque}^2 \tau_{disque}^2}{4\pi^2 l_{fils}} \tag{3.7}$$

Avec τ_{disque} la période d'oscillation du disque. Si l'on attache un objet, de moment d'inertie sur l'axe c I_{objet}, de masse m_{objet} et de période d'oscillation τ_{objet}, au disque on obtient une inertie totale de :

$$I_{disque} + I_{objet} = \frac{(m_{disque} + m_{objet}) g r_{disque}^2 \tau_{objet}^2}{4\pi^2 l_{fils}} \tag{3.8}$$

Si l'on utilise une référence, où son moment d'inertie I_{ref}, sa masse m_{ref} et la période d'oscillation τ_{ref} sont connues, on peut remarquer que :

$$I_{ref} = \frac{(m_{disque} + m_{ref}) g r_{disque}^2 \tau_{ref}^2}{4\pi^2 l_{fils}} - I_{disque} \tag{3.9}$$

$$= \frac{g r_{disque}^2}{4\pi^2 l_{fils}} \left((m_{disque} + m_{ref}) \tau_{ref}^2 - m_{disque} \tau_{disque}^2 \right) \tag{3.10}$$

On peut alors trouver, à partir de (3.8) et (3.10) :

$$I_{objet} = \frac{g r_{disque}^2}{4\pi^2 l_{fils}} \left((m_{disque} + m_{objet}) \tau_{objet}^2 - m_{disque} \tau_{disque}^2 \right) \tag{3.11}$$

$$= \frac{I_{ref} (m_{disque} + m_{objet}) \tau_{objet}^2 - m_{disque} \tau_{disque}^2}{(m_{disque} + m_{ref}) \tau_{ref}^2 - m_{disque} \tau_{disque}^2} \tag{3.12}$$

$$= I_{ref} \frac{\left(1 + \frac{m_{objet}}{m_{disque}}\right) \frac{\tau_{objet}^2}{\tau_{disque}^2} - 1}{\left(1 + \frac{m_{ref}}{m_{disque}}\right) \frac{\tau_{ref}^2}{\tau_{disque}^2} - 1} \tag{3.13}$$

Il est alors possible de trouver le moment d'inertie d'un objet sur l'axe étant le même que c à partir d'un objet de référence. On peut, par exemple, récupérer un objet de forme cubique étant composé d'une seule matière dont la masse est connue. Les moments d'inertie sur l'axe du centre de gravité sont alors simples à calculer et peuvent être utilisés comme une référence.

On a utilisé, pour calculer les moments d'inertie I_{xx}, I_{yy} et I_{zz} du drone, une horloge permettant de chronométrer le temps au quadcopter à faire dix oscillations afin de pouvoir agrandir la précision. L'expérience a été réalisée trois fois sur un même axe afin de réaliser une moyenne.

Figure 3.4 Mesure du moment d'inertie I_{xx}

Figure 3.5 Mesure du moment d'inertie I_{yy}

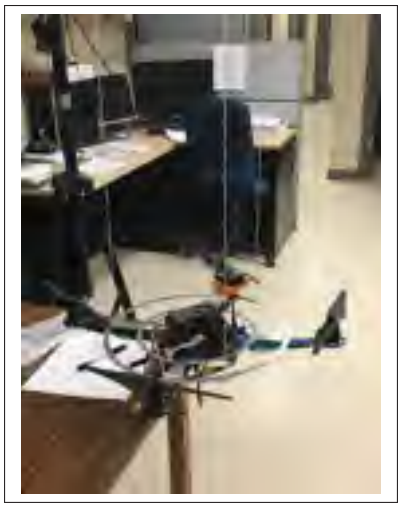

Figure 3.6 Mesure du moment d'inertie I_{zz}

Le tableau 3.2 présente les différents résultats des moments d'inertie obtenus à partir de cette méthode.

Tableau 3.2 Données de la mesure des moments d'inertie

	Référence	**Disque**	**Drone pour I_{xx}**	**Drone pour I_{yy}**	**Drone pour I_{zz}**
Masse (kg)	0,2408	0,0908	1,354	1,354	1,354
Période (s)	2,708	2,68	1,856	1,85	2,452
Inertie (kg.m^2)	0,0052	inconnue	0,0126	0,0125	0,0235

On peut voir que nous avons des résultats proches de la méthode théorique. Cette méthode fonctionne correctement lorsque la masse du drone est largement supérieure avec celle du disque. Ceci n'est cependant pas le cas lorsque l'on veut mesurer les moments d'inertie d'un micro drone. La masse de ce dernier étant négligeable par rapport au disque, la technique employée ne peut pas être utilisée. On a alors recours à la méthode se basant sur un modèle en 3D.

3.2.3 Méthode par modélisation 3D

La précision de la méthode experimentale dépend de plusieurs facteurs : la précision au niveu du relevé de la periode d'oscillation, le placement du drone sous le disque, la présence d'oscil-

lations non voules. Une solution à ce problème est d'utiliser un logiciel de modélisation 3D. On peut alors modéliser chaque pièce du drone ainsi que donner une masse aux pièces. Le logiciel calculera alors les moments d'inertie totaux lorsque l'on assemblera les pièces. On concevra un modèle 3D du drone S500 sur le logiciel Solidworks (Dassault Systèmes (2017)) (Figure 3.7). Le tableau 3.3 montre les moments d'inertie pour le drone.

Tableau 3.3 Moments d'inertie par Solidworks du drone S500

I_{xx} (kg · m^2)	I_{yy} (kg · m^2)	I_{zz} (kg · m^2)
0,012 75	0,012 78	0,022 71

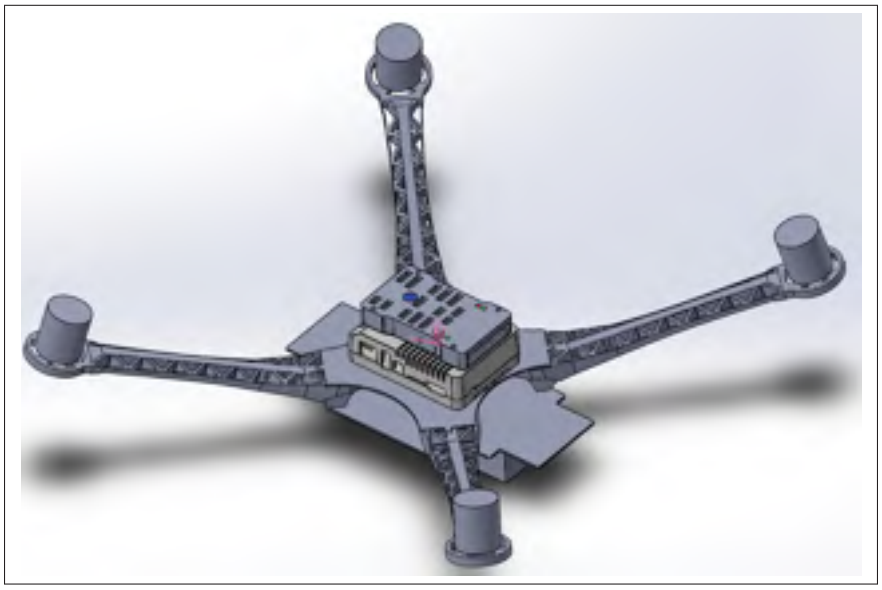

Figure 3.7 Modèle 3D du drone S500

3.3 Relation entre les forces et les signaux MLI

On a vu dans la section précédente les méthodes pouvant être utilisées afin de trouver les moments d'inertie. La commande du drone enverra en sortie les couples selon l'orientation du drone ainsi que la force de portance. On utilisera alors ces résultats afin de déterminer les

forces exercées par chaque moteur (voir la relation (2.44). Les forces ne sont cependant pas directement contrôlables en pratique. La carte de commande envoie en sortie des signaux MLI faisant varier les vitesses des moteurs. On a besoin de trouver une méthode afin de déterminer la relation entre les signaux MLI ainsi que les forces.

Le drone S500 utilise des moteurs sans balais munis de contrôlleurs de vitesse. Le signal MLI envoyé par la carte de commande ne sert alors pas à obtenir une période cyclique pour la tension du moteur. Le contrôleur de vitesse dispose d'un microprocesseur permettant de créer à partir du signal MLI trois signaux déphasés de 120 degrés pour alimenter le moteur. Il devient alors difficile d'établir une relation directe entre le signal MLI envoyé et la tension efficace du moteur pour contrer la chute de tension de la batterie. On utilise dans un premier temps un dynamomètre de série 1580 de RCbenchmark (Figure 3.8). Cet appareil de mesure dispose en plus de quatre sorties MLI, d'un accéléromètre, d'un voltmètre, d'un ampèremètre et permet de mesurer la vitesse du moteur.

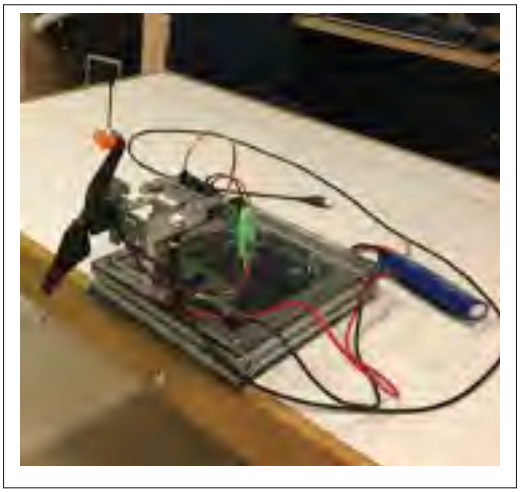

Figure 3.8 Mesure de la force

On récupère alors la force exercée par le moteur en fonction de la largeur d'impulsion en μs. On obtient le résultat suivant (Figure 3.9) :

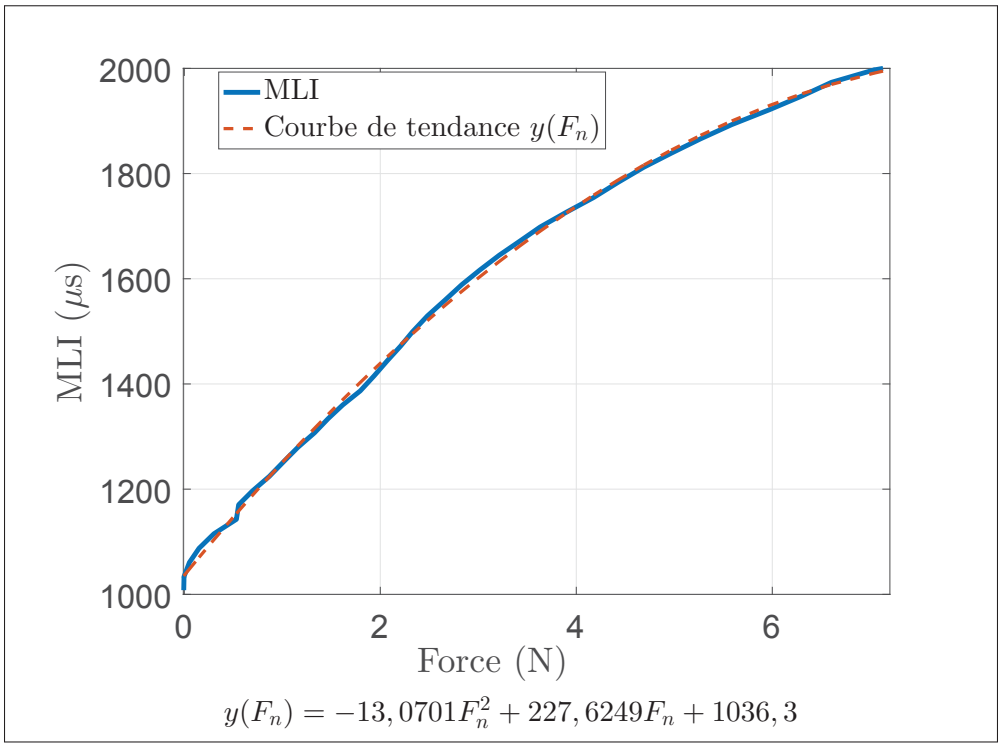

Figure 3.9 MLI en fonction de la force du moteur

On peut alors, à l'aide de la courbe de tendance, déterminer une équation permettant de relier la force d' un moteur. On trouve :

$$Largeur\ d'impulsion(\mu s) = -13{,}0701F_n^2 + 227{,}6249F_n + 1036{,}3 \tag{3.14}$$

On veut trouver la relation entre le couple et la force d'un moteur. On utilise alors la mesure suivante (Figure 3.10) :

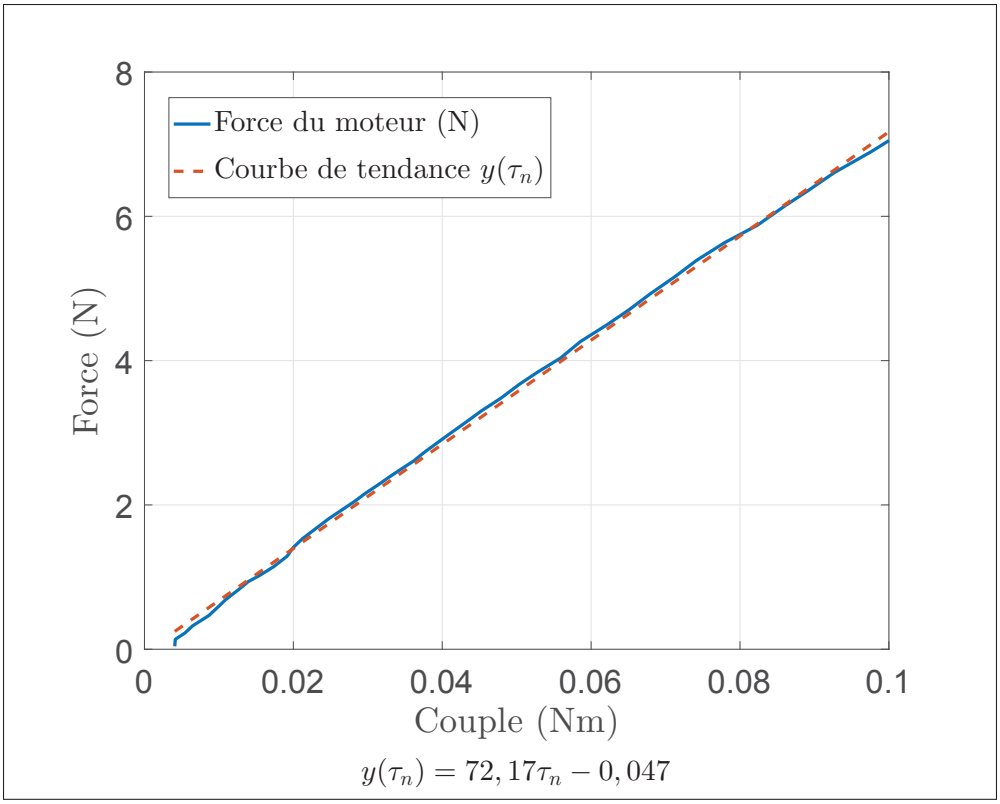

Figure 3.10 Relation entre la force et le couple d'un moteur

On peut alors, à l'aide d'une régression linéaire, obtenir une courbe de tendance décrite par :

$$F_n = 72{,}17\tau_n - 0{,}047 \tag{3.15}$$

Il est important de noter que ces relevées ont été pris lorsque la batterie était complètement chargée. La tension de la batterie chute en effet en fonction du temps. Les résultats indiqués ci-dessus ne correspondront alors plus à la réalité. On décidera de modéliser la chute de tension par une force externe fictive qui perturberait le drone. La loi adaptative sera alors utilisée pour corriger ce problème.

3.4 Conclusion

On a vu tout au long de ce chapitre trois méthodes permettant de récupérer les moments d'inertie. On utilisera les valeurs des moments d'inertie données par la méthode expérimentale pour

le drone S500 lors de l'implémentation. On a determiné ensuite la relation entre la force et les signaux MLI afin de pouvoir correctement commander les drones. La relation a été trouvée à l'aide du dynamomètre 1580 de RCbenchmark. Les paramètres du drone sont à présent connus. On peut concevoir la commande.

CHAPITRE 4

COMMANDE DU QUADCOPTER

4.1 Structure de la commande

On contrôlera le drone en deux parties. Un contrôleur d'attitude gérant l'orientation du drone sera mis en place. Ce contrôleur aura pour entrées les angles de rotation (roulis, lacet et tangage) ainsi que la force de portance désirée et en sortie les couples nécessaires à la rotation du drone. On pourra alors utiliser les couples et la force de portance pour retrouver les forces nécessaires des 4 moteurs en utilisant la relation (2.44). Les angles de rotation désirés ainsi que la force de portance seront retrouvés grâce au contrôleur de position dans lequel les positions voulues du drone seront en entrée (voir Figure 4.1).

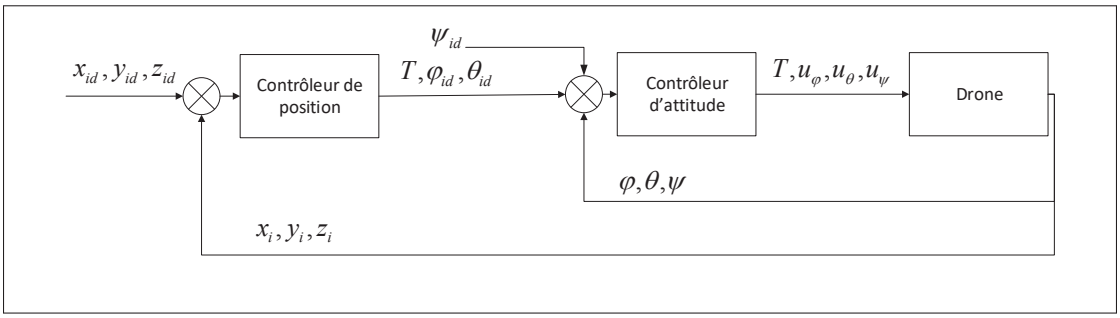

Figure 4.1 Structure de la commande du drone

Afin que les contrôleurs fonctionnent correctement, il est impératif que le contrôleur d'attitude s'exécute plus rapidement que le contrôleur de position. Il faut de plus que ce contrôleur se mette à jour rapidement (de l'ordre d'une centaine de Hertz). Le contrôleur d'attitude doit en effet pouvoir changer les vitesses des moteurs avant qu'une nouvelle information sur les angles désirés soit venue. De plus, les contrôleurs doivent pouvoir faire leurs opérations plus rapidement que l'obtention d'une nouvelle information venant des différents capteurs de position et d'angle. Tout ceci doit être pris en compte lors de la réalisation du contrôleur en pratique.

4.2 Linéarisation du modèle et conception des contrôleurs du drone

La dynamique du drone est fortement couplée et non linéaire comme on peut le voir dans la relation (2.33). Il est donc nécessaire de linéariser le modèle afin de pouvoir concevoir nos contrôleurs. Le but de la linéarisation est de trouver une variable d'état représentant le drone de la forme :

$$\dot{\boldsymbol{Z}} = \boldsymbol{A}\boldsymbol{Z} + \boldsymbol{B}\boldsymbol{u}_{ent} \tag{4.1}$$

On séparera la variable d'état montrée dans (2.33) en deux parties afin de concevoir le contrôleur d'attitude et de position.
On utilisera la variable d'état suivante pour traiter le contrôleur d'attitude :

$$\dot{\boldsymbol{\Theta}} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} p + q\sin(\phi)\tan(\theta) + r\cos(\phi)\tan(\theta) \\ q\cos(\phi) - r\sin(\phi) \\ q\sin(\phi)\sec(\theta) + r\cos(\phi)\sec(\theta) \\ (I_{yy} - I_{zz})qr/I_{xx} + u_{\phi}/I_{xx} \\ (I_{zz} - I_{xx})pr/I_{yy} + u_{\theta}/I_{yy} \\ (I_{xx} - I_{yy})pq/I_{zz} + u_{\psi}/I_{zz} \end{bmatrix} \tag{4.2}$$

Le contrôleur de position utilisera la variable d'état ci-dessous :

$$\dot{\boldsymbol{X}}_{pos} = \begin{bmatrix} \ddot{x}_i \\ \dot{x}_i \\ \ddot{y}_i \\ \dot{y}_i \\ \ddot{z}_i \\ \dot{z}_i \end{bmatrix} = \begin{bmatrix} {}^{i}F_x/m - \Delta_x \\ \dot{x}_i \\ {}^{i}F_y/m - \Delta_y \\ \dot{y}_i \\ {}^{i}F_z/m - g - \Delta_z \\ \dot{z}_i \end{bmatrix} \tag{4.3}$$

4.2.1 Linéarisation entrée-sortie et conception du contrôleur d'attitude

Il est important de définir quelques notations avant de linéariser la variable décrite dans l'équation (4.2). On appelle $L_f h(x)$ la dérivée de Lie de h dans la direction $f(x)$ la relation suivante :

$$L_f h(x) = \frac{\partial h}{\partial x} f(x) \tag{4.4}$$

On note les dérivées i fois successives $L_f^i h(x)$ de Lie par rapport à $f(x)$:

$$L_f^i h(x) = L_f(L_f^{i-1} h(x)) \tag{4.5}$$

La linéarisation de la variable d'état (4.2) consiste à faire un changement de variable afin de retrouver une forme décrite dans l'équation (4.1). En posant :

$$\boldsymbol{y}_a = \boldsymbol{h}_a(\boldsymbol{\Theta}) \tag{4.6}$$

$$\dot{\boldsymbol{\Theta}} = \boldsymbol{f}_a(\boldsymbol{\Theta}) + \boldsymbol{G}_a(\boldsymbol{\Theta})\boldsymbol{u} \tag{4.7}$$

Avec :

$$\boldsymbol{h}_a(\Theta) = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T \tag{4.8}$$

$$\boldsymbol{f}_a(\Theta) = \begin{bmatrix} p + q\sin(\phi)\tan(\theta) + r\cos(\phi)\tan(\theta) \\ q\cos(\phi) - r\sin(\phi) \\ q\sin(\phi)\sec(\theta) + r\cos(\phi)\sec(\theta) \\ (I_{yy} - I_{zz})qr/I_{xx} \\ (I_{zz} - I_{xx})pr/I_{yy} \\ (I_{xx} - I_{yy})pq/I_{zz} \end{bmatrix} \tag{4.9}$$

$$\boldsymbol{G}_a(\boldsymbol{\Theta}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1/I_{xx} & 0 & 0 \\ 0 & 1/I_{yy} & 0 \\ 0 & 0 & 1/I_{zz} \end{bmatrix} \tag{4.10}$$

On peut remarquer que :

$$\dot{\boldsymbol{y}}_a = \frac{\partial \boldsymbol{h}_a}{\partial \boldsymbol{\Theta}} \dot{\boldsymbol{\Theta}} \tag{4.11}$$

$$= \frac{\partial \boldsymbol{h}_a}{\partial \boldsymbol{\Theta}} \boldsymbol{f}_a(\boldsymbol{\Theta}) + \frac{\partial \boldsymbol{h}_a}{\partial \boldsymbol{\Theta}} \boldsymbol{G}_a(\boldsymbol{\Theta}) \boldsymbol{u}_\Theta \tag{4.12}$$

$$= \boldsymbol{L}_{f_a} \boldsymbol{h}_a + \boldsymbol{L}_{g_a} \boldsymbol{h}_a \cdot \boldsymbol{u}_\Theta \tag{4.13}$$

Il est facilement démontrable que $L_{g_a} h_a = 0$. On obtient par conséquent une relation ne dépendant pas des entrées du système. On doit alors dériver une seconde fois y_a. On obtient alors :

$$\ddot{\boldsymbol{y}}_a = \frac{\partial \boldsymbol{L}_{f_a} \boldsymbol{h}_a}{\partial \boldsymbol{\Theta}} \dot{\boldsymbol{\Theta}} \tag{4.14}$$

$$= \frac{\partial \boldsymbol{L}_{f_a} \boldsymbol{h}_a}{\partial \boldsymbol{\Theta}} \boldsymbol{f}_a + \frac{\partial \boldsymbol{L}_{f_a} \boldsymbol{h}_a}{\partial \boldsymbol{\Theta}} \boldsymbol{G}_a \cdot \boldsymbol{u}_\Theta \tag{4.15}$$

$$= \boldsymbol{L}_{f_a}^{\boldsymbol{2}} \boldsymbol{h}_a + \boldsymbol{L}_{g_a} \boldsymbol{L}_{f_a} \boldsymbol{h}_a \cdot \boldsymbol{u}_\Theta \tag{4.16}$$

On peut démontrer que $L_{g_a} L_{f_a} h_a \neq 0$. On peut alors établir une nouvelle variable d'état $\boldsymbol{z}_a = \begin{bmatrix} \boldsymbol{z}_{a1} & \boldsymbol{z}_{a2} \end{bmatrix}$ définie par :

$$\boldsymbol{z}_{a1} = \boldsymbol{y}_a \tag{4.17}$$

$$\dot{\boldsymbol{z}}_{a1} = \dot{\boldsymbol{y}}_a \tag{4.18}$$

$$\dot{\boldsymbol{z}}_{a2} = \ddot{\boldsymbol{y}}_a \tag{4.19}$$

Soit v_a une loi de commande linéaire. On peut alors définir u tel que :

$$\boldsymbol{u}_\Theta = \frac{\boldsymbol{v} - \boldsymbol{L}_{f_a}^2 \boldsymbol{h}_a}{\boldsymbol{L}_{g_a} \boldsymbol{L}_{f_a} \boldsymbol{h}_a} \tag{4.20}$$

En injectant u_Θ dans (4.16), on retrouve $\dot{z}_{a2} = v$. N'importe quelle commande linéaire peut être alors utilisée pour le drone. On prendra la loi de commande suivante :

$$\boldsymbol{v} = \begin{bmatrix} v_\phi & v_\theta & v_\psi \end{bmatrix}^T \tag{4.21}$$

$$\boldsymbol{v} = \ddot{\boldsymbol{\Theta}}_d + k_{\Theta p} \cdot \boldsymbol{E}_\Theta + k_{\Theta d} \cdot \dot{\boldsymbol{E}}_\Theta \tag{4.22}$$

Avec $\ddot{\Theta}_d$ les accélérations désirées des angles d'orientation, $E_\Theta = \Theta_d - \Theta$ l'erreur entre les positions des angles désirées Θ_d et les angles actuels, $\dot{E}_\Theta = \dot{\Theta}_d - \dot{\Theta}$ l'erreur entre les vitesses désirées $\dot{\Theta}_d$ et les vitesses des angles actuelles et $k_{\Theta p}$ et $k_{\Theta d}$ des constantes positives. En développant (4.20) on obtient :

$$u_\phi = I_{xx} \left(\frac{\sin(2\phi) r^2}{2} + v_\phi - \frac{q^2 \sin(2\phi)}{2} - v_\psi \sin(\theta) - qr\cos(2\phi) \right) + qr\left(-I_{yy} + I_{zz}\right) \tag{4.23}$$

$$\begin{aligned} u_\theta = & I_{xx} pr + I_{yy} \left(pr + v_\theta \cos(\phi) + q^2 \cos(\phi)^3 \tan(\theta) - r^2 \cos(\phi)^3 \tan(\theta) + v_\psi \cos(\theta) \sin(\phi) \right. \\ & \left. - q^2 \cos(\phi) \tan(\theta) + 2r^2 \cos(\phi) \tan(\theta) + \frac{3qr \sin(\phi) \tan(\theta)}{2} - \frac{qr \sin(3\phi) \tan(\theta)}{2} \right) \\ & - I_{zz} pr \end{aligned} \tag{4.24}$$

$$\begin{aligned} u_\psi = & -I_{xx} pq + I_{yy} pq + I_{zz} \left[v_\psi \cos(\phi) \cos(\theta) - v_\theta \sin(\phi) - \tan(\theta) \left(\sin(\phi) q^2 \cos(\phi)^2 \right. \right. \\ & \left. \left. + \sin(\phi) q^2 + 2qr \cos(\phi)^3 - \sin(\phi) r^2 \cos(\phi)^2 \right) - pq \right] \end{aligned} \tag{4.25}$$

On peut voir que chaque couple dépend de plusieurs lois de commande. Les expressions précédentes peuvent être difficiles à manipuler lors de la conception d'une commande. Une solution à ce problème est d'étudier la dynamique du drone lorsque les angles d'orientation ϕ et θ sont au voisinage de zéro. On peut alors approximer l'expression (4.9) par :

$$\boldsymbol{f}_a(\Theta) = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ (I_{yy} - I_{zz})\dot{\theta}\dot{\psi}/I_{xx} \\ (I_{zz} - I_{xx})\dot{\phi}\dot{\psi}/I_{yy} \\ (I_{xx} - I_{yy})\dot{\phi}\dot{\theta}/I_{zz} \end{bmatrix} \tag{4.26}$$

En utilisant la procédure faite précédemment on trouve les couples suivants :

$$u_\phi = I_{xx} v_\phi + (I_{zz} - I_{yy})\dot{\psi}\dot{\theta} \tag{4.27}$$

$$u_\theta = I_{yy} v_\theta + (I_{xx} - I_{zz})\dot{\phi}\dot{\psi} \tag{4.28}$$

$$u_\psi = I_{zz} v_\psi + (I_{yy} - I_{xx})\dot{\phi}\dot{\theta} \tag{4.29}$$

Cette approximation peut être réalisée que pour des trajectoires et manœuvres du drone particulières. Nous utiliserons des trajectoires simples où les angles d'orientation ϕ et θ du drone seront toujours au voisinage de zéro. Nous pourrons donc utiliser cette approche.

4.2.2 Conception du contrôleur de position

On souhaite commander le drone en position en utilisant la variable d'état (4.3). On peut cependant voir que nous avons des forces Δ_x, Δ_y et Δ_z dont leurs valeurs sont inconnues. Les perturbations sont considérées comme étant bornées et variant peu au cours du temps. On doit concevoir une commande adaptative.

On s'intéresse dans un premier temps à contrôler l'altitude du drone à savoir la variable z_i.

On pose :

$$e_z = z_{id} - z_i \tag{4.30}$$

$$\dot{e}_z = \dot{z}_{id} - \dot{z}_i \tag{4.31}$$

$$\ddot{e}_z = \ddot{z}_{id} - \ddot{z}_i \tag{4.32}$$

Avec z_{id} la position désirée en z_i du drone. On peut alors en déduire à partir de (4.3) :

$$\ddot{e}_z = \ddot{z}_{id} + g + \Delta_z - {}^i F_z/m \tag{4.33}$$

Si Δ_z était connue, ${}^iF_z$ pourrait avoir la valeur :

$${}^iF_z = m(v_z + g + \ddot{z}_{id} + \Delta_z) \tag{4.34}$$

$$v_z = k_p e_z + k_d \dot{e}_z \tag{4.35}$$

Avec k_p et k_d des constantes positives. On aurait alors, en injectant (4.34) dans (4.33), une dynamique de l'erreur de la forme :

$$\dot{\boldsymbol{E}}_z = \boldsymbol{A} \cdot \boldsymbol{E}_z \tag{4.36}$$

$$\begin{bmatrix} \dot{e}_z \\ \ddot{e}_z \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_p & -k_d \end{bmatrix} \begin{bmatrix} e_z \\ \dot{e}_z \end{bmatrix} \tag{4.37}$$

Si on calcule les gains k_p et k_d de sorte que la matrice A soit Hurwitz, l'erreur e_z convergera de manière exponentielle à zéro. On ne connait cependant pas Δ_z. On pose alors :

$$\tilde{\Delta}_z = \Delta_z - \hat{\Delta}_z \tag{4.38}$$

Avec $\hat{\Delta}$ l'estimation de Δ_z. On peut alors poser à partir de (4.33)

$$\ddot{e}_z = \ddot{z}_{id} + g + \Delta_z - \hat{\Delta}_z + \hat{\Delta}_z - {}^i F_z/m \tag{4.39}$$

$$\ddot{e}_z = \ddot{z}_{id} + g + \tilde{\Delta}_z + \hat{\Delta}_z - {}^i F_z/m \tag{4.40}$$

En posant :

$${}^i F_z = m(v_z + g + \ddot{z}_{id} + \hat{\Delta}_z) \tag{4.41}$$

$$v_z = k_p e_z + k_d \dot{e}_z \tag{4.42}$$

$$\dot{\hat{\Delta}}_z = \frac{e_z \Gamma_z}{2k_p} + \frac{\dot{e}_z \Gamma_z (k_p + 1)}{2k_p k_d} \tag{4.43}$$

On obtient une loi de commande où l'erreur e_z et $\tilde{\Delta}_z$ tendent vers zéro si Γ_z est une constante positive. Afin de démontrer la convergence de l'erreur à zéro, on pose la dynamique de l'erreur sous la forme :

$$\ddot{e}_z = -v_z + \tilde{\Delta}_z \tag{4.44}$$

$$\begin{bmatrix} \dot{e}_z \\ \ddot{e}_z \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_p & -k_d \end{bmatrix} \begin{bmatrix} e_z \\ \dot{e}_z \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tilde{\Delta}_z \tag{4.45}$$

$$\dot{\boldsymbol{E}}_z = \boldsymbol{A}\boldsymbol{E}_z + \boldsymbol{\Psi}\tilde{\Delta}_z \tag{4.46}$$

On peut voir dans l'équation (4.46) que si $\tilde{\Delta}_z$ tend vers zéro, la dynamique de l'erreur dispose d'une matrice Hurwitz. Le système est donc stable en boucle fermée. On veut à présent vérifier la stabilité au sens de Lyapunov du contrôleur ainsi que la loi d'adaptation. On décide d'utiliser la fonction de Lyapunov suivante :

$$V = \boldsymbol{E}_z^T \boldsymbol{P}\boldsymbol{E}_z + \Gamma_z^{-1}\tilde{\Delta}_z^2 \tag{4.47}$$

Avec P une matrice symétrique définie positive telle que $A^T P + PA = -Q$, Q étant une matrice symétrique définie positive. La dérivée de la fonction de Lyapunov devient alors :

$$\dot{V} = \dot{\boldsymbol{E}}_z^T \boldsymbol{P}\boldsymbol{E}_z + \boldsymbol{E}_z^T \boldsymbol{P}\dot{\boldsymbol{E}}_z + 2\dot{\tilde{\Delta}}_z \Gamma_z^{-1} \tilde{\Delta}_z \tag{4.48}$$

$$= \boldsymbol{E}_z^T (\boldsymbol{A}^T \boldsymbol{P} + \boldsymbol{P}\boldsymbol{A})\boldsymbol{E}_z + 2\tilde{\Delta}_z (\boldsymbol{\Psi}^T \boldsymbol{P}\boldsymbol{E}_z + \Gamma_z^{-1} \dot{\tilde{\Delta}}_z) \tag{4.49}$$

$$= -\boldsymbol{E}_z^T \boldsymbol{Q}\boldsymbol{E}_z + 2\tilde{\Delta}_z (\boldsymbol{\Psi}^T \boldsymbol{P}\boldsymbol{E}_z + \Gamma_z^{-1} \dot{\tilde{\Delta}}_z) \tag{4.50}$$

Pour avoir une fonction $\dot{V}$ définie semi-négative, la condition suivante doit être respectée :

$$\boldsymbol{\Psi}^T \boldsymbol{P}\boldsymbol{E}_z = -\Gamma_z^{-1} \dot{\tilde{\Delta}}_z \tag{4.51}$$

$$\dot{\tilde{\Delta}}_z = -\Gamma_z \boldsymbol{\Psi}^T \boldsymbol{P}\boldsymbol{E}_z \tag{4.52}$$

Si Δ_z varie peu dans le temps nous avons alors :

$$\dot{\tilde{\Delta}}_z = -\dot{\tilde{\Delta}}_z \tag{4.53}$$

On se retrouve avec une fonction $\dot{V}$ définie semi-négative. On ne peut cependant pas confirmer la convergence de l'erreur à zéro. Afin de prouver la convergence de l'erreur, le lemme de Barbalat peut être utilisé (voir Annexe II). On peut alors prouver la convergence de l'erreur à zéro en vérifiant que la fonction $\dot{V}$ est uniformément continue. En dérivant $\dot{V}$ et en appliquant les conditions décrites on obtient :

$$\ddot{V} = -2\dot{\boldsymbol{E}}_z^T \boldsymbol{Q}\boldsymbol{E}_z \tag{4.54}$$

On peut noter que la variable E_z est bornée puisque $V \geq 0$ et que $\tilde{\Delta}_z$ est bornée. On peut alors, à partir de (4.46), démontrer que $\dot{E}_z$ est aussi bornée. $\dot{V}$ est alors uniformément continue et l'erreur converge donc à zéro.

En utilisant la relation $A^T P + PA = -Q$ et en prenant Q comment étant la matrice identité de dimension 2 on retrouve :

$$\boldsymbol{P} = \begin{bmatrix} P_{11} & P_{12} \\ P_{12} & P_{22} \end{bmatrix} \tag{4.55}$$

$$P_{11} = \frac{k_d^2 + k_p^2 + k_p}{2k_p k_d} \tag{4.56}$$

$$P_{12} = \frac{1}{2k_p} \tag{4.57}$$

$$P_{22} = \frac{k_p + 1}{2k_p k_d} \tag{4.58}$$

On peut alors déduire à partir de (4.52) et (4.53) :

$$\dot{\hat{\Delta}}_z = \frac{e_z \Gamma_z}{2k_p} + \frac{\dot{e}_z \Gamma_z (k_p + 1)}{2k_p k_d} \tag{4.59}$$

${}^iF_x$ et ${}^iF_y$ peuvent être retrouvées de la même manière que ${}^iF_z$. On se retrouve alors avec les expressions :

$$\dot{\hat{\Delta}}_x = \frac{e_x \Gamma_x}{2k_p} + \frac{\dot{e}_x \Gamma_x (k_p + 1)}{2k_p k_d} \tag{4.60}$$

$$\dot{\hat{\Delta}}_y = \frac{e_y \Gamma_y}{2k_p} + \frac{\dot{e}_y \Gamma_y (k_p + 1)}{2k_p k_d} \tag{4.61}$$

$$\dot{\hat{\Delta}}_z = \frac{e_z \Gamma_z}{2k_p} + \frac{\dot{e}_z \Gamma_z (k_p + 1)}{2k_p k_d} \tag{4.62}$$

$${}^iF_x = m(k_p e_x + k_d \dot{e}_x + \ddot{x}_{id} + \hat{\Delta}_x) \tag{4.63}$$

$${}^iF_y = m(k_p e_y + k_d \dot{e}_y + \ddot{y}_{id} + \hat{\Delta}_y) \tag{4.64}$$

$${}^iF_z = m(k_p e_z + k_d \dot{e}_z + g + \ddot{z}_{id} + \hat{\Delta}_z) \tag{4.65}$$

Le contrôleur d'attitude demande en entrée la force de portance ainsi que les angles. En se basant sur la relation (2.34) on peut en déduire :

$$^{i}F_x = T(\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)) \tag{4.66}$$

$$^{i}F_y = T(\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi)) \tag{4.67}$$

$$^{i}F_z = T\cos(\phi)\cos(\theta) \tag{4.68}$$

Si on pose l'angle ψ comme étant connu, on peut alors résoudre le système d'équations décrit par (4.66), (4.67) et (4.68) :

$$T = \sqrt{^{i}F_x^2 + ^{i}F_y^2 + ^{i}F_z^2} \tag{4.69}$$

$$\phi_d = \arcsin\left(\frac{^{i}F_x\sin(\psi) - ^{i}F_y\cos(\psi)}{T}\right) \tag{4.70}$$

$$\theta_d = \arctan\left(\frac{^{i}F_x\cos(\psi) + ^{i}F_y\sin(\psi)}{^{i}F_z}\right) \tag{4.71}$$

On peut noter que la force de portance T est supérieure à zéro comme le montre l'équation (2.39). L'angle ϕ_d est alors toujours défini. La force $^{i}F_z$ devient nulle lorsque ϕ ou θ est égal à $\pm\frac{\pi}{2}$ radian. Ces valeurs d'angle ne seront cependant jamais atteintes afin d'éviter le blocage de cadran. L'angle de lacet ψ serait remplacé par ψ_d lors de l'implémentation.

4.3 Conclusion

On a vu que le contrôleur du drone est composé en deux parties : une partie dite contrôleur d'attitude permet de contrôler l'orientation du drone. La deuxième partie du contrôleur s'occupe de la position du drone dans l'espace. On connait à présent les lois de commande et d'adaptation. On peut utiliser le contrôleur si l'on connait certains paramètres :

a. Les positions actuelles du drone ;

b. Les vitesses actuelles du drone ;

c. Les angles d'orientation du drone ;

d. Les inerties et la masse du drone.

Le chapitre suivant traitera de la récupération des positions et vitesses actuelles du drone.

CHAPITRE 5

DÉTECTION DE LA POSITION ET ESTIMATION DE LA VITESSE DU DRONE AVEC LA KINECT2

5.1 Modélisation de la Kinect V2

La commande de position doit avoir en entrée les positions et vitesses actuelles du drone (Figure 4.1). La Kinect V2 est alors utilisée pour récupérer la position. La Kinect permet de récupérer la distance entre la caméra et l'environnement. On peut récupérer, à partir de la caméra, une matrice I_a de dimension (512,424) où chaque élément est une valeur de 16 bits indiquant la distance entre les objets pris par la Kinect et la caméra en millimètre.

On peut, à partir d'un algorithme, récupérer l'élément de coordonnée (u_{cdrone}, v_{cdrone}) de la matrice où le drone se situe. La valeur de cet élément sera alors la coordonnée x_i si la Kinect est orientée horizontalement. La matrice ne permet cependant pas de directement récupérer les coordonnées y_i et z_i. Il faut trouver une relation entre les coordonnées (u_{cdrone}, v_{cdrone}) et les coordonnées (x_i, y_i, z_i). On utilise pour cela le modèle du sténopé.

5.1.1 Modèle du sténopé

Le modèle du sténopé (Hartley & Zisserman (2003)) se base sur une représentation simple d'une caméra. Il est caractérisé par plusieurs paramètres dits intrinsèques et d'autres, dits extrinsèques. Les paramètres intrinsèques dépendent exclusivement des caractéristiques de la caméra (distorsion, longueur focale...). Les paramètres extrinsèques dépendent de la position de la caméra sur l'environnement (position et orientation de l'appareil). Il est alors évident que les paramètres intrinsèques ne changeront pas au cours du temps. Les paramètres extrinsèques devront être modifiés à chaque changement de position ou d'orientation de la caméra.

Plusieurs repères doivent être pris en comptes afin de modéliser correctement la caméra (Figure 5.1). Le repère $\{i\}$ est le repère inertiel où se trouve les coordonnées en mètres x_i, y_i et z_i. Le repère$\{f\}$ est le repère dit rétinien. Ce repère est à une distance$\{f_o\}$ (dit longueur focale) du plan de l'image. L'ouverture de l'objectif se situe sur l'origine de ce repère. Le repère $\{k\}$

représente le repère en millimètres du plan de l'image. Son origine se situe au point principal P_p qui se trouve au milieu du plan. Le repère $\{c\}$ est le repère du plan de l'image en pixel. Son origine se situe au bord du plan.

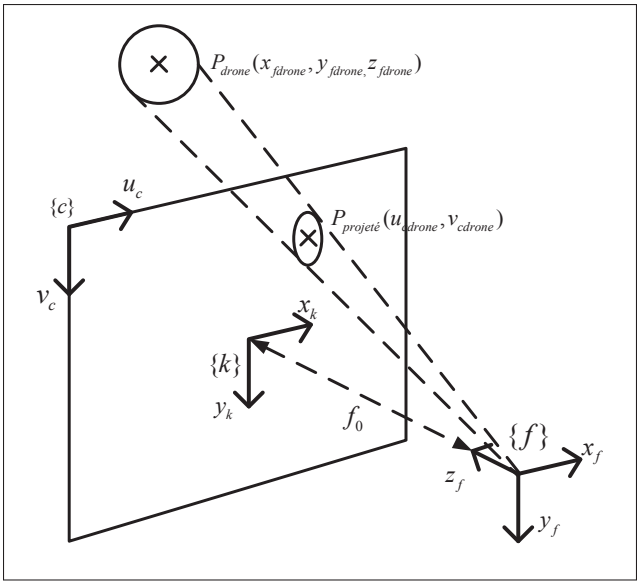

Figure 5.1 Schéma du modèle du sténopé

La relation entre le repère $\{f\}$ et $\{k\}$ peut être trouvée à partir du théorème de Thales (voir Figure 5.2).

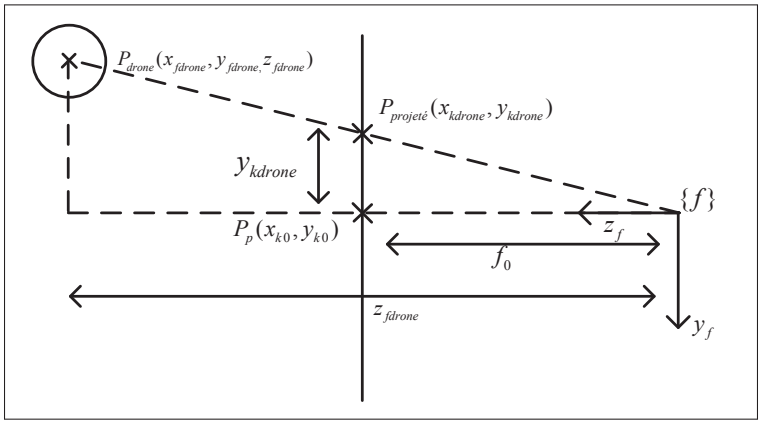

Figure 5.2 Vue de côté du plan de l'image

On peut alors en déduire :

$$x_{kdrone} = \frac{f_0 x_{fdrone}}{z_{fdrone}} \tag{5.1}$$

$$y_{kdrone} = \frac{f_0 y_{fdrone}}{z_{fdrone}} \tag{5.2}$$

Les relations (5.1) et (5.2) peuvent être écrites en utilisant une matrice de transformation homogène :

$$\lambda \cdot \begin{bmatrix} x_{kdrone} \\ y_{kdrone} \\ 1 \end{bmatrix} = \begin{bmatrix} f_0 & 0 & 0 & 0 \\ 0 & f_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{fdrone} \\ y_{fdrone} \\ z_{fdrone} \\ 1 \end{bmatrix} \tag{5.3}$$

Avec $\lambda = \frac{1}{z_f}$. La matrice de transformation pour passer du repère $\{k\}$ au repère $\{c\}$ est décrite de la manière suivante :

$$^{c}_{k}\boldsymbol{T} = \begin{bmatrix} k_u & -k_v \cot(\theta_u) & u_{c0} - \cot(\theta_u) v_{c0} \\ 0 & \frac{k_v}{\sin(\theta_u)} & \frac{v_{c0}}{\sin(\theta)} \\ 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

Avec k_u et k_v des gains en pixels par millimètres permettant de faire la relation entre la distance parcourue pour un pixel donné dans la direction u_c et v_c. θ_u représente l'angle entre les axes du repère $\{c\}$. Le repère peut, en effet, ne pas être orthogonal dû à l'image de la caméra. On peut néanmoins utiliser une approximation de cet angle le mettant égal à $\frac{\pi}{2}$. On se retrouve alors avec :

$$^{c}_{k}\boldsymbol{T} = \begin{bmatrix} k_u & 0 & u_{c0} \\ 0 & k_v & v_{c0} \\ 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

On peut, à partir des relations (5.3) et (5.4), trouver la relation entre les coordonnées $(x_{fdrone}, y_{fdrone}, z_{fdrone})$ et (u_{cdrone}, v_{cdrone}). On obtient :

$$\begin{bmatrix} u_{cdrone} \\ v_{cdrone} \\ 1 \end{bmatrix} = \frac{1}{\lambda} \begin{bmatrix} f_0 k_u & -f_0 k_v \cot(\theta_u) & u_{c0} - \cot(\theta_u) v_{c0} & 0 \\ 0 & f_0 \frac{k_v}{\sin(\theta_u)} & \frac{v_{c0}}{\sin(\theta)} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{fdrone} \\ y_{fdrone} \\ z_{fdrone} \\ 1 \end{bmatrix} \tag{5.6}$$

On utilise souvent dans la littérature la notation suivante pour la matrice homogène :

$$\begin{bmatrix} u_{cdrone} \\ v_{cdrone} \\ 1 \end{bmatrix} = \frac{1}{\lambda} \begin{bmatrix} \alpha_0 & s & \alpha_1 & 0 \\ 0 & \beta_0 & \beta_1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{fdrone} \\ y_{fdrone} \\ z_{fdrone} \\ 1 \end{bmatrix} \tag{5.7}$$

Les constantes α_0, α_1, β_0, β_1 et s décrivent les paramètres intrinsèques de la camera. Ces constantes peuvent être données par le constructeur (le SDK de la Kinect donne ces informations (disponible sur https://msdn.microsoft.com/en-us/library/) ou peuvent être trouvées en faisant une série de calibrage (Herrera *et al.* (2011)). La valeur de λ sera trouvée grâce au capteur de profondeur de la Kinect. La relation entre le repère $\{f\}$ et le repère inertiel $\{i\}$ du drone peut se faire à l'aide d'une matrice de transformation homogène. Cette matrice dépendra de l'emplacement de la Kinect par rapport au drone et de l'emplacement de l'origine du repère $\{i\}$.

5.2 Détermination de la position du drone

On peut, à partir de (5.7), retrouver une coordonnée dans le repère inertiel en millimètre pour un pixel donné dans le repère $\{c\}$. On doit cependant traiter l'image reçue de sorte à pouvoir isoler le drone de l'environnement. On pourra alors récupérer le centre du drone comme point de référence afin de recevoir une coordonnée unique dans le repère $\{c\}$. On peut utiliser, pour isoler le drone, un algorithme appelé soustraction de l'arrière-plan (*Background subtraction*

dans la littérature (Sobral & Vacavant (2014))). Le principe de cet algorithme est démontré dans la Figure 5.3.

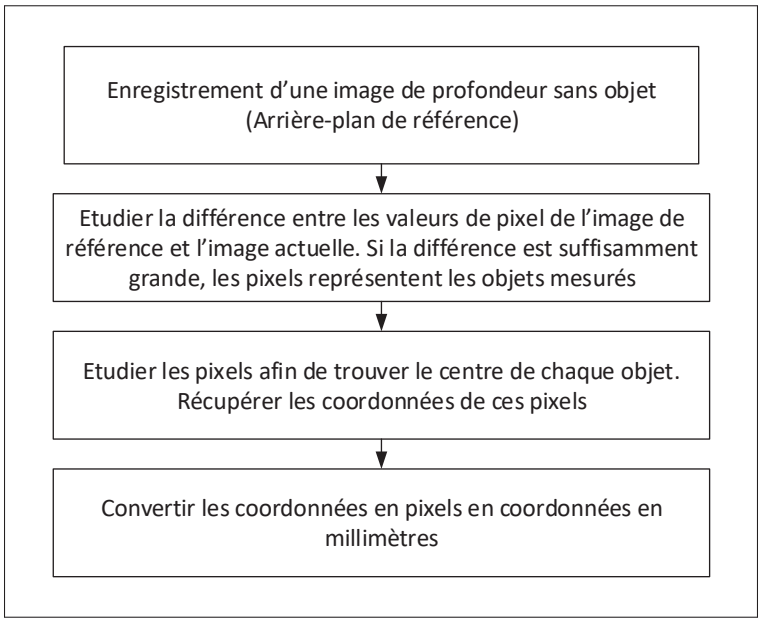

Figure 5.3 Soustraction de l'arrière plan

La Figure 5.4 présente un example d'image de référence à savoir un arrière plan. On peut alors, une fois l'image de référence générée, mettre l'objet dont la position doit être mesurée devant la caméra (Figure 5.5). La Figure 5.6 montre le résultat de l'algorithme.

Figure 5.4 Image de référence

Figure 5.5 Image avec objets à mesurer

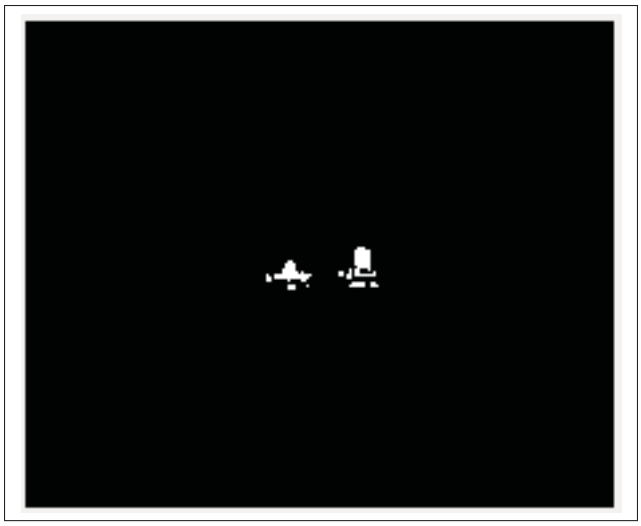

Figure 5.6 Soustraction de l'arrière-plan

Cet algorithme fonctionne sous plusieurs conditions. L'environnement et la caméra doivent être fixes pendant toute l'expérimentation. Les objets n'appartenant pas à l'environnement doivent être des objets dont l'on souhaite mesurer. Les objets doivent être suffisamment grands pour ne pas être confondus avec du bruit. Des recherches sur ce domaine ont été menées afin de corriger ces problèmes (arrière-plan dynamique, prise en compte de plusieurs caméras...). Les solutions complexifient cependant les algorithmes et peuvent augmenter le temps de calcul de la position. L'utilisation de l'algorithme classique pour notre projet est suffisante étant donné l'environnement utilisé pour les expérimentations (enclos fermé dans un laboratoire). Une synthèse des différentes techniques sur la soustraction de l'arrière-plan a été faite par Bouwmans (2014).

On utilise, pour réaliser la soustraction de l'arrière-plan, la matrice I_a générée par la Kinect. Afin de faciliter les calculs, il peut être préférable de normaliser les images de la profondeur pour avoir des valeurs comprises entre 0 et 1. La matrice I_a génère du bruit. Une portion du bruit peut être filtrée facilement en plaçant les valeurs en dessous de 0.01 à 1 lorsque l'image est normalisée (Algorithme 5.1).

Algorithme 5.1 Normalisation de l'image de profondeur

```
   Entrées : Image actuelle Ia de la profondeur
   Sorties : Image normalisée et filtrée
 1 début
 2     lire Ia;
 3     Ia = Ia / Profondeur max;
 4     pour chaque élément de Ia faire
 5         si élément actuel de Ia>1 alors
 6             élément actuel de Ia=1;
 7         fin
 8         sinon si élément actuel de Ia<0.01 alors
 9             élément actuel de Ia=1;
10         fin
11     fin
12     retourner I_a
13 fin
```

On peut choisir une profondeur maximale arbitraire selon l'attente de l'expérimentation. Ceci permet notamment de ne pas traiter des objets étant dans le champ de vision de la Kinect, mais n'étant pas dans la zone d'expérimentation. Du bruit produit par les capteurs de la Kinect peut modifier les valeurs des éléments de la matrice de profondeur. On risque alors de comparer l'image actuelle avec du bruit, on aura alors produit une mauvaise image de référence. Pour pallier à ce problème, la génération de l'arrière-plan ne se basera pas sur une image, mais sur plusieurs (Algorithme 5.2). Le nombre d'images doit être suffisamment grand pour avoir un résultat optimal (une centaine d'images doit être produite).

Algorithme 5.2 Génération de l'arrière-plan normalisé

Entrées : Image normalisée *Ia*
Sorties : Arrière-plan normalisé *Ib*

```
1  début
2      lire Ia;
3      initialiser Ib en prenant Ib=Ia;
4      tant que le nombre d'images à scanner n'est pas atteint faire
5          pour chaque nouvelle image de Ia faire
6              pour chaque élément de Ia et Ib faire
7                  prendre la valeur minimale entre l'élément de Ia et celui de Ib;
8                  remplacer l'élément de Ib par la valeur minimale calculée;
9              fin
10         fin
11     fin
12     retourner I_b
13 fin
```

La différence entre l'image actuelle et l'arrière-plan peut alors être faite. Il est nécessaire d'utiliser des images binaires pour traiter l'image. La matrice finale aura alors une valeur de 1. On choisira alors de mettre à 1 les éléments de la matrice dont la valeur est supérieure à une certaine limite et à 0 si ce n'est pas le cas. Cette limite est déterminée arbitrairement. Elle doit être suffisamment élevée pour pouvoir différencier les objets du bruit, mais suffisamment faible pour différencier les objets à l'arrière-plan. Du bruit peut tout de même exister (Figure 5.7). On utilise alors la morphologie d'ouverture qui consiste à faire une érosion entre l'image générée et une forme géométrique (ici un carré contenant de dimension 3 pixels par 3 pixels) suivi d'une dilation (Figure 5.6). Une explication plus poussée peut être lue en suivant Dougherty & Lotufo (2003).

Figure 5.7 Image sans ouvrage

On peut à présent identifier les centres des objets recherchés afin de récupérer leurs coordonnées. On peut utiliser pour cela une transformation de distance dans l'image binaire. Cette méthode permet d'obtenir une matrice désignant pour chaque élément la distance euclidienne au carrée en pixel entre le pixel actuel et le contour de l'objet (Figure 5.8)

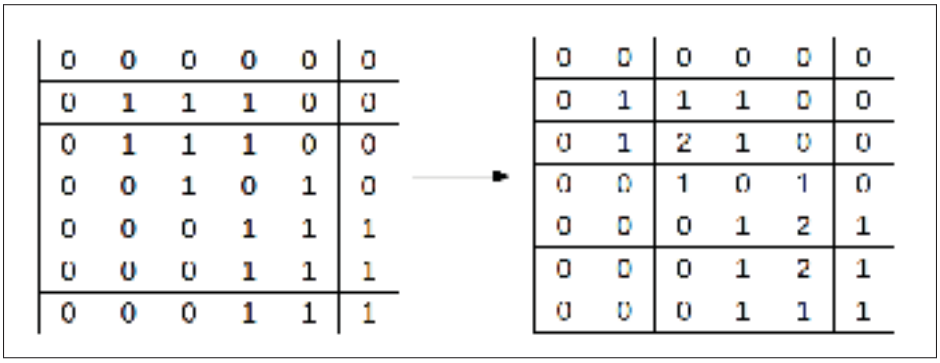

Figure 5.8 Exemple d'une transformation de distance

On peut voir dans la Figure 5.8 que les valeurs les plus élevées se situent au centre des objets (représentés par des 1 dans la première matrice). On peut cependant voir qu'on obtient deux pixels d'une valeur de 2 pour un des objets. Le milieu ne pouvant pas être défini entre deux pixels la première valeur sera prise. On obtient trois valeurs élevées pour deux objets. On a

besoin d'identifier les éléments de la matrice qui représentent un objet. On utilise pour cela un algorithme dit de remplissage par diffusion. Cet algorithme permet, à partir d'un pixel de départ (appelé graine) de vérifier les valeurs des pixels voisins (on considère un pixel voisin comme étant les quatre pixels ayant en commun un bord avec le pixel d'origine). Si au moins un des pixels voisins n'a pas une valeur nulle, les valeurs des pixels sont égales à 1. On continue de faire le même procédé aux pixels voisins ayant une valeur non nulle. On s'arrête lorsqu'on ne peut plus trouver de nouveaux voisins ayant une valeur non nulle. On met alors tous les autres pixels à 0. La Figure 5.9 illustre le résultat. La case orange est la graine.

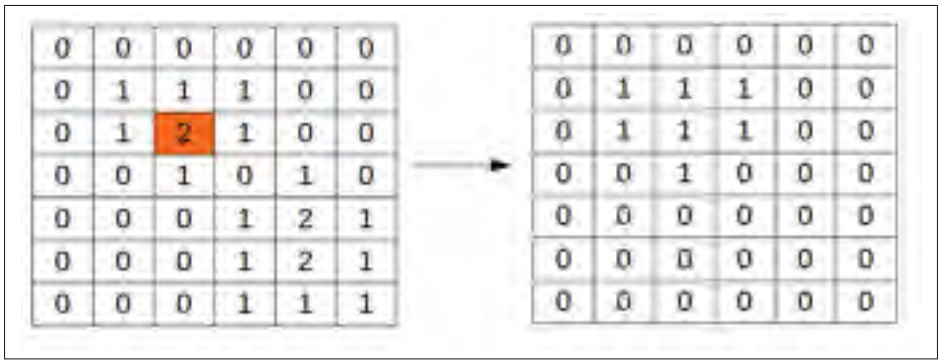

Figure 5.9 Exemple de remplissage par diffusion

On se retrouve alors avec une matrice pouvant être utilisée comme masque. On peut alors utiliser ce masque comme le montre l'Algorithme 5.3 afin de récupérer les centres de plusieurs objets.

Algorithme 5.3 Récupération de la position de plusieurs objets à partir de la Kinect

```
   Entrées : Image Ia
   Sorties : Position en millimètres des objets
1  début
2      lire Ia;
3      normaliser Ia;
4      générer l'arrière-plan normalisé Ib;
5      pour chaque nouvelle image de Ia faire
6          prendre les valeurs minimales entre Ib et ia;
7          faire la différence entre Ib et Ia;
8          faire un ouvrage de la matrice calculée;
9          mettre à 1 si la différence est supérieure à la limite, 0 sinon;
10         faire une transformation de distance, prendre les coordonnées en pixel de
             l'élément contenant la valeur maximale du résultat précédent;
11         tant que valeur maximale >0 faire
12             convertir les coordonnées en millimètres, sauvegarder la position;
13             faire un masque à l'aide de l'algorithme de remplissage;
14             soustraire l'image binaire par le masque;
15             faire une transformation de distance sur la nouvelle matrice;
16             prendre les coordonnées en pixel de l'élément contenant la valeur maximale
                 du résultat précédent;
17         fin
18         convertir les coordonnées en millimètre;
19     fin
20 fin
```

On a vu que plusieurs coordonnées en millimètres peuvent être récupérées grâce à la Kinect. Les algorithmes précédents permettent de réduire le bruit produit par la caméra. On peut cependant avoir du bruit dû à de la transformation de distance. On a aussi besoin de générer la vitesse du drone pour notre contrôleur. On a besoin d'un filtre de position ainsi qu'un estimateur de vitesse.

5.3 Estimation de la vitesse

Une solution directe d'avoir la vitesse est de dériver la position actuelle du drone en faisant le calcul :

$$\dot{x}[k] \approx \frac{x[k] - x[k-1]}{T_{ech}} \tag{5.8}$$

Avec T_{ech} le temps de rafraîchissement de la Kinect. Cette solution pose cependant un problème. La position génère du bruit (Figure 5.10). On peut donc soustraire une valeur totalement erronée. On se retrouve facilement à avoir une vitesse bruitée comme le montre la Figure 5.11.

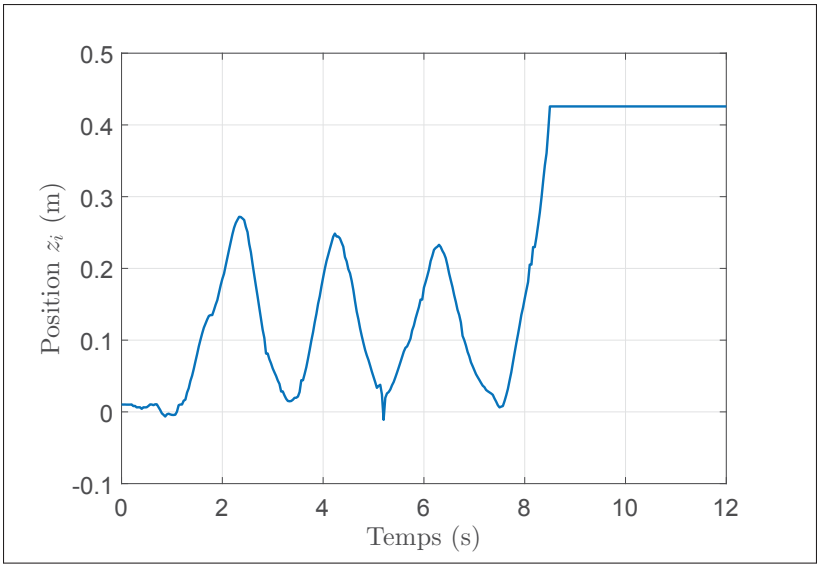

Figure 5.10 Position actuelle bruitée en z_i du drone

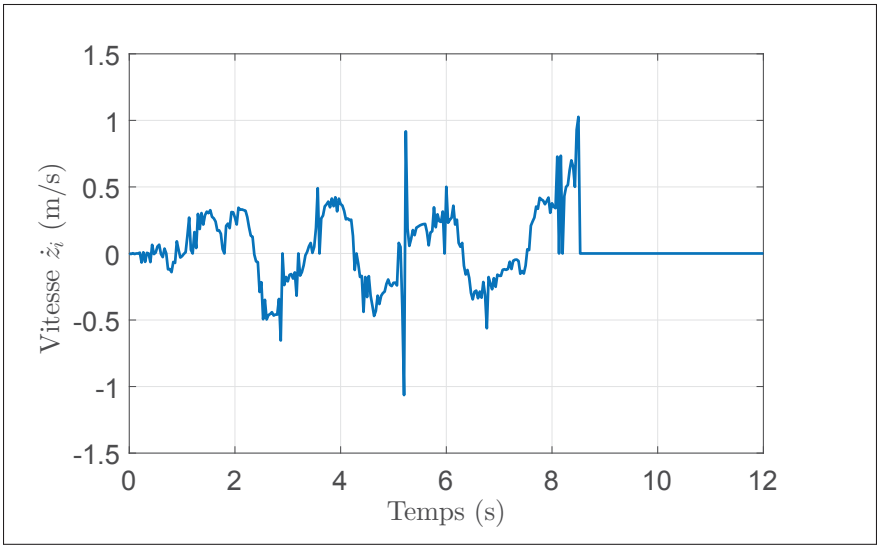

Figure 5.11 Estimation de la vitesse actuelle $\dot{z}_i$ du drone en utilisant la dérivé directe

Deux solutions seront proposées : la première utilisera un filtre passe-bas de deuxième ordre et la deuxième utilisera le filtre de Kalman.

5.3.1 Estimation de la vitesse à l'aide d'un filtre passe-bas de deuxième ordre

L'estimation de la vitesse avec un filtre passe-bas de deuxième ordre consiste à utiliser les propriétés du filtre pour récupérer la vitesse. Soit $u_{raw}(t)$ le signal bruité et $y_{lp}(t)$ la sortie du filtre. Le filtre passe-bas du second ordre est décrit par :

$$\frac{Y_{lp}(s)}{U_{raw}(s)} = \frac{1}{\frac{s^2}{\omega_r^2} + \frac{2\zeta s}{\omega_r} + 1} \tag{5.9}$$

Avec $Y_{lp}(s)$ et $U_{raw}(s)$ les signaux respectives en Laplace de la sortie et de l'entrée du filtre, ζ le taux d'amortissement et ω_r la pulsation de résonance . En considérant que $y_{lp}(0) = 0$ on peut déduire à partir de la transformée de Laplace :

$$\mathscr{L}\{\dot{y}_{lp}(t)\}(s) = s \cdot Y_{lp}(s) \tag{5.10}$$

$$\mathscr{L}\{\dot{y}_{lp}(t)\}(s) = \frac{s \cdot U_{raw}(s)}{\frac{s^2}{\omega_r^2} + \frac{2\zeta s}{\omega_r} + 1} \tag{5.11}$$

On peut alors utiliser la relation (5.11) pour récupérer la vitesse et (5.9) pour filtrer la position. Le filtre doit cependant pouvoir être implémenté. La forme actuelle du filtre ne permet pas l'implémentation. Notre système est en effet discret ce qui n'est pas le cas avec le filtre décrit précédemment. On doit trouver l'équivalent du filtre passe-bas en discret. On considère alors le signal U^*_{raw} comme étant la discrétisation de U_{Raw} pour un temps d'échantillonnage T_{ech}. On considère le schéma suivant (Figure 5.12) :

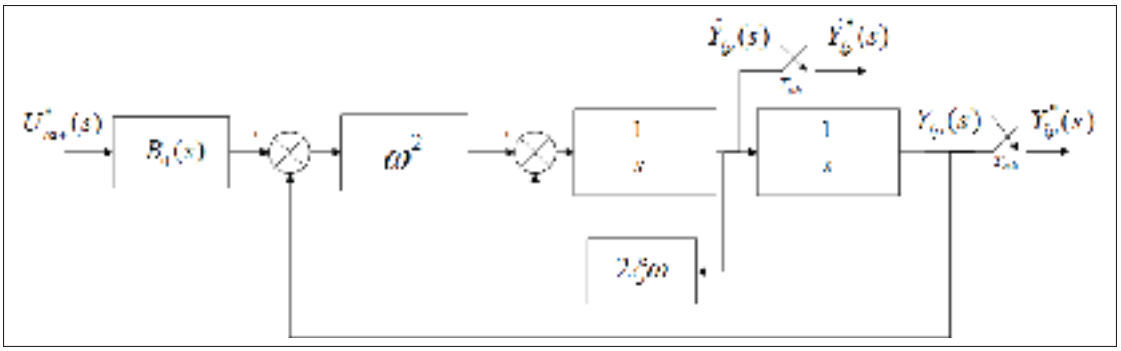

Figure 5.12 Schéma bloc du passe-bas de deuxième ordre discrétisé

Afin de pouvoir utiliser le filtre passe-bas de deuxième ordre, un bloqueur d'ordre zéro $B_o(s)$ est implémenté. Le bloqueur d'ordre zéro peut être modélisé par :

$$B_0(s) = \frac{1 - \mathrm{e}^{-st}}{s} \tag{5.12}$$

En prenant compte du schéma décrit par la Figure 5.12 on se retrouve avec une sortie discrète Y_{lp}^* de la forme :

$$Y_{lp}^*(s) = \frac{B_0(s)}{\frac{s^2}{\omega_r^2} + \frac{2\zeta s}{\omega_r} + 1} U_{raw}^*(s) \tag{5.13}$$

$$\dot{Y}_{lp}^*(s) = \frac{s \cdot B_0(s)}{\frac{s^2}{\omega_r^2} + \frac{2\zeta s}{\omega_r} + 1} U_{raw}^*(s) \tag{5.14}$$

On a besoin d'avoir une équation par récurrence pour les deux sorties. On doit utiliser la transformée en Z. On peut à partir de (5.12), (5.13) et (5.14) et les règles de la transformé en Z déduire :

$$\mathscr{Z}\{Y_{lp}^*(s)\}(z) = (1 - z^{-1})\mathscr{Z}\left\{\frac{Y_{lp}(s)}{s}\right\}(z) \tag{5.15}$$

$$\mathscr{Z}\{\dot{Y}_{lp}^*(s)\}(z) = (1 - z^{-1})\mathscr{Z}\left\{\frac{\dot{Y}_{lp}(s)}{s}\right\}(z) \tag{5.16}$$

On obtient alors une équation de la forme :

$$\mathscr{Z}\left\{\frac{Y^*_{lp}(s)}{U^*_{raw}(s)}\right\}(z) = \frac{a_1 z + a_2}{z^2 + b_1 z + b_2} \tag{5.17}$$

$$\mathscr{Z}\left\{\frac{\dot{Y}^*_{lp}(s)}{U^*_{raw}(s)}\right\}(z) = \frac{a_3 z + a_4}{z^2 + b_1 z + b_2} \tag{5.18}$$

On peut à partir de là obtenir les équations de récurrence pour le filtre de la position et de la vitesse en posant :

$$y^*_{lp}[k] = a_1 u^*_{raw}[k-1] + a_2 u^*_{raw}[k-2] - b_1 y^*_{lp}[k-1] - b_2 y^*_{lp}[k-2] \tag{5.19}$$

$$\dot{y}^*_{lp}[k] = a_3 u^*_{raw}[k-1] + a_4 u^*_{raw}[k-2] - b_1 \dot{y}^*_{lp}[k-1] - b_2 \dot{y}^*_{lp}[k-2] \tag{5.20}$$

Les valeurs des coefficients a_1, a_2, a_3, a_4, b_1 et b_2 sont donnés dans l'Annexe I. La valeur de ζ doit être comprise entre $\frac{\sqrt{2}}{2}$ et 1 pour que le filtre fonctionne correctement. La valeur de ω_r doit être assez faible pour filtrer les harmoniques de hautes fréquences et suffisamment élevées pour ne pas filtrer le signal utile. La Figure 5.13 montre le résultat du filtre en mettant une valeur de ζ de 0,9 et une pulsation ω_r de $2\pi 6$ rad/s. La Figure 5.14 montre l'effet du filtre sur la position.

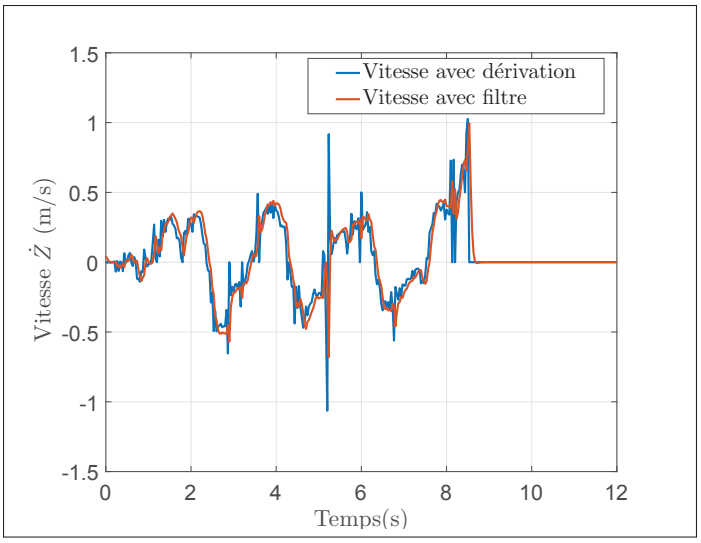

Figure 5.13 Estimation de la vitesse actuelle $\dot{z}_i$ du drone en utilisant le filtre passe-bas

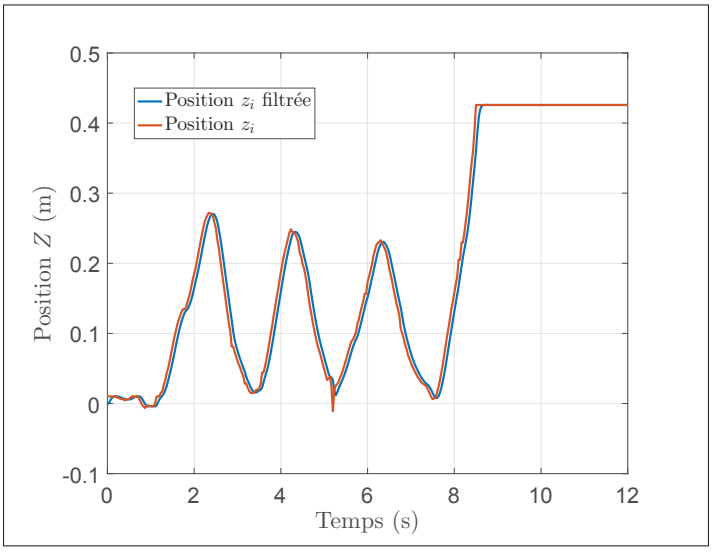

Figure 5.14 Position z_i avec le filtre passe-bas

Le filtre passe-bas peut être utilisé si le signal d'origine a du faible bruit. Le filtre passe-bas peut s'avérer être assez lent lorsque l'on essaie de filtrer beaucoup plus que ce qu'on a fait précédemment. Ceci se traduit par un décalage du signal par rapport au signal d'origine. Pour régler ce problème, un filtre de Kalman a été utilisé à la place du passe-bas de deuxième ordre.

5.3.2 Estimation de la vitesse à l'aide d'un filtre de Kalman discret

On veut utiliser le filtre Kalman comme un observateur où on peut récupérer les vitesses du drone à partir de positions bruitées et de manière discrète. On peut modéliser l'estimation de la variable d'état X_i bruitée par une variable d'état discrète X_{kal} décrite par :

$$\boldsymbol{X}_{kal} = \begin{bmatrix} \hat{x}_i & \hat{y}_i & \hat{z}_i & \dot{\hat{x}}_i & \dot{\hat{y}}_i & \dot{\hat{z}}_i \end{bmatrix}^T \tag{5.21}$$

$$\boldsymbol{X}_{kal}[k+1] = F_{kal} \cdot X_{kal}[k] + G_{kal} \cdot w[k] \tag{5.22}$$

$$\boldsymbol{Y}_{kal}[k] = C_{kal} \cdot X_{kal}[k] + v_{kal}[k] \tag{5.23}$$

$$\boldsymbol{F}_{kal} = \begin{bmatrix} 1 & 0 & 0 & T_{ech} & 0 & 0 \\ 0 & 1 & 0 & 0 & T_{ech} & 0 \\ 0 & 0 & 1 & 0 & 0 & T_{ech} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.24}$$

$$\boldsymbol{C}_{kal} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{5.25}$$

$$\boldsymbol{G}_{kal} = \begin{bmatrix} T_{ech}/2 & 0 & 0 \\ 0 & T_{ech}/2 & 0 \\ 0 & 0 & T_{ech}/2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.26}$$

Où $w_{kal}[k]$ et $v_{kal}[k]$ représentent respectivement un vecteur de bruit blanc venant du procédé et de la mesure. Ces bruits ont une distribution gaussienne définie par une espérance nulle et une matrice de covariance R_{kal} pour $v_{kal}[k]$ et Q_{kal} pour $w_{kal}[k]$. $v_{kal}[k]$ et $w_{kal}[k]$ sont considérés comme indépendants entre eux. Y_{kal} décrit les variables mesurables. Les variables d'estimation $\dot{\hat{x}}_i$, $\dot{\hat{y}}_i$ et $\dot{\hat{z}}_i$ sont modélisées comme étant des valeurs aléatoires $w_{kal}[k]$ autour d'une constante soit :

$$\dot{\hat{x}}_i[k+1] = \dot{\hat{x}}_i[k] + w_{kal}(1,1)[k] \tag{5.27}$$

$$\dot{\hat{y}}_i[k+1] = \dot{\hat{y}}_i[k] + w_{kal}(2,1)[k] \tag{5.28}$$

$$\dot{\hat{z}}_i[k+1] = \dot{\hat{z}}_i[k] + w_{kal}(3,1)[k] \tag{5.29}$$

Les variables d'estimation des positions $\hat{x}_i$, $\hat{y}_i$ et $\hat{z}_i$ sont modélisés par :

$$\frac{\hat{x}_i[k+1] - \hat{x}_i[k]}{T_{ech}} = \frac{\dot{\hat{x}}_i[k+1] + \dot{\hat{x}}_i[k]}{2} \tag{5.30}$$

$$\frac{\hat{y}_i[k+1] - \hat{y}_i[k]}{T_{ech}} = \frac{\dot{\hat{y}}_i[k+1] + \dot{\hat{y}}_i[k]}{2} \tag{5.31}$$

$$\frac{\hat{z}_i[k+1] - \hat{z}_i[k]}{T_{ech}} = \frac{\dot{\hat{z}}_i[k+1] + \dot{\hat{z}}_i[k]}{2} \tag{5.32}$$

Les équations (5.30), (5.31) et (5.32) décrient une approximation de la dérivée de la position en discret. Ce modèle se base sur des vitesses ayant des valeurs aléatoires, mais dont la moyenne de ces vitesses pour un temps d'échantillonnage équivaut à la dérivée discrète de la position. On peut à partir de là concevoir un observateur permettant de récupérer les vitesses à partir des positions mesurées. En considérant l'estimation de la variable d'état sans le bruit :

$$\hat{\boldsymbol{X}}^{-}_{obs}[k+1] = \boldsymbol{F}_{kal} \cdot \hat{\boldsymbol{X}}^{-}_{obs}[k] \tag{5.33}$$

On peut établir un observateur de la forme :

$$\hat{\boldsymbol{X}}_{obs}[k+1] = \boldsymbol{F}_{kal} \cdot \hat{\boldsymbol{X}}^{-}_{obs}[k] + \boldsymbol{K}_{kal}(\boldsymbol{Y}_{kal}[k] - \boldsymbol{C}_{kal}\hat{\boldsymbol{X}}^{-}_{obs}[k]) \tag{5.34}$$

Les variables d'état $\hat{X}^{-}_{obs}$ et $\hat{X}_{obs}$ sont souvent appelées respectivement la variable d'état *a priori* et la variable d'état *a posteriori*. La variable d'état $\hat{X}^{-}_{obs}$ estime les variables avant d'effectuer la mesure. La variable d'état $\hat{X}_{obs}$ estime les variables une fois la mesure prise en compte. Un schéma bloc est montré dans la Figure 5.15.

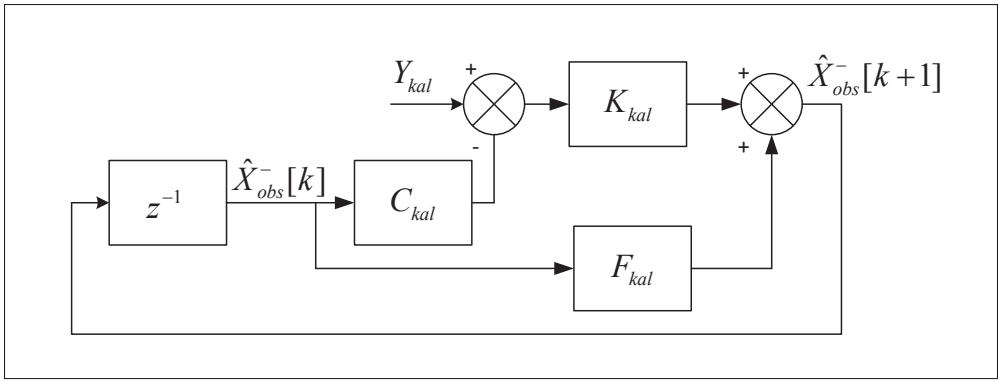

Figure 5.15 Schéma bloc du filtre de Kalman

On peut alors récupérer les estimations des vitesses à l'aide de cet observateur. Le but du filtre de Kalman est de trouver une valeur de K_{kal} de manière à optimiser le résultat de l'observateur en considérant les covariances des bruits de mesure et du procédé. On considère les erreurs suivantes :

$$\boldsymbol{e}_{kal}[k] = \boldsymbol{X}_{kal} - \hat{\boldsymbol{X}}_{obs} \tag{5.35}$$

$$\boldsymbol{e}^{-}_{kal}[k] = \boldsymbol{X}_{kal} - \hat{\boldsymbol{X}}^{-}_{obs} \tag{5.36}$$

Le gain K_{kal} doit être optimisé de sorte à minimiser la covariance P_{kal} de l'erreur *a posteriori* soit :

$$\boldsymbol{P}_{kal} = \mathrm{E}[\boldsymbol{e}_{kal}\boldsymbol{e}^{T}_{kal}] \tag{5.37}$$

Plusieurs manières sont utilisées pour minimiser P_{kal}. Grover & Hwang (2012) et Simon (2006) proposent différentes solutions pour calculer K_{kal}. On décide de prendre la solution :

$$\boldsymbol{K}_{kal}[k] = \boldsymbol{F}_{kal}\boldsymbol{P}_{kal}[k]\boldsymbol{C}_{kal}^{T}(\boldsymbol{C}_{kal}\boldsymbol{P}_{kal}[k]\boldsymbol{C}_{kal}^{T} + \boldsymbol{R}_{kal})^{-1} \tag{5.38}$$

$$\boldsymbol{M}_{kal}[k] = \boldsymbol{P}_{kal}[k]\boldsymbol{C}_{kal}^{T}(\boldsymbol{C}_{kal}\boldsymbol{P}_{kal}[k]\boldsymbol{C}_{kal}^{T} + \boldsymbol{R}_{kal})^{-1} \tag{5.39}$$

$$\boldsymbol{Z}_{kal}[k] = (\boldsymbol{I} - \boldsymbol{M}_{kal}[k]\boldsymbol{C}_{kal})\boldsymbol{P}_{kal}[k](\boldsymbol{I} - \boldsymbol{M}_{kal}[k]\boldsymbol{C}_{kal})^{T} + \boldsymbol{M}_{kal}[k]\boldsymbol{R}_{kal}\boldsymbol{M}_{kal}^{T}[k] \tag{5.40}$$

$$\boldsymbol{P}_{kal}[k+1] = \boldsymbol{F}_{kal}\boldsymbol{Z}_{kal}[k]\boldsymbol{F}_{kal}^{T} + \boldsymbol{G}_{kal}\boldsymbol{Q}_{kal}\boldsymbol{G}_{kal}^{T} \tag{5.41}$$

On peut voir le comportement du gain en fonction des bruits des signaux. Si la mesure ne dispose pas de bruit, la relation (5.34) est alors égale à $F_{kal} \cdot X_{kal}$ l'observateur ne prendra alors que les valeurs mesurées. En revanche si le procédé n'est pas bruité, la relation (5.34) sera alors égale à $F_{kal} \cdot \hat{X}_{obs}^{-}[k]$. La difficulté revient à trouver les matrices des covariances des bruits $w_{kal}[k]$ et $v_{kal}[k]$. On peut considérer que les bruits sont indépendants entre eux.
On peut alors estimer R_{kal} par la matrice :

$$\boldsymbol{R}_{kal} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix} \tag{5.42}$$

Avec σ_x^2, σ_y^2 et σ_z^2 les variances de $\hat{x}_i$, $\hat{y}_i$ et $\hat{z}_i$. Ces variances peuvent être obtenues facilement en calculant les variances des signaux lorsque le drone est immobile. La matrice Q_{kal} ne peut cependant pas être mesurée facilement. Des mesures empiriques doivent être faites pour obtenir Q_{kal}. En prenant $Q_{kal} = diag\left(\begin{bmatrix} 8 \cdot 10^{-4} & 8 \cdot 10^{-4} & 8 \cdot 10^{-4} \end{bmatrix}\right)$ et σ_x^2, σ_y^2 et σ_z^2 égales à $2 \cdot 10^{-6}$ on obtient les résultats décrits par la Figures 5.16 et la Figure 5.17.

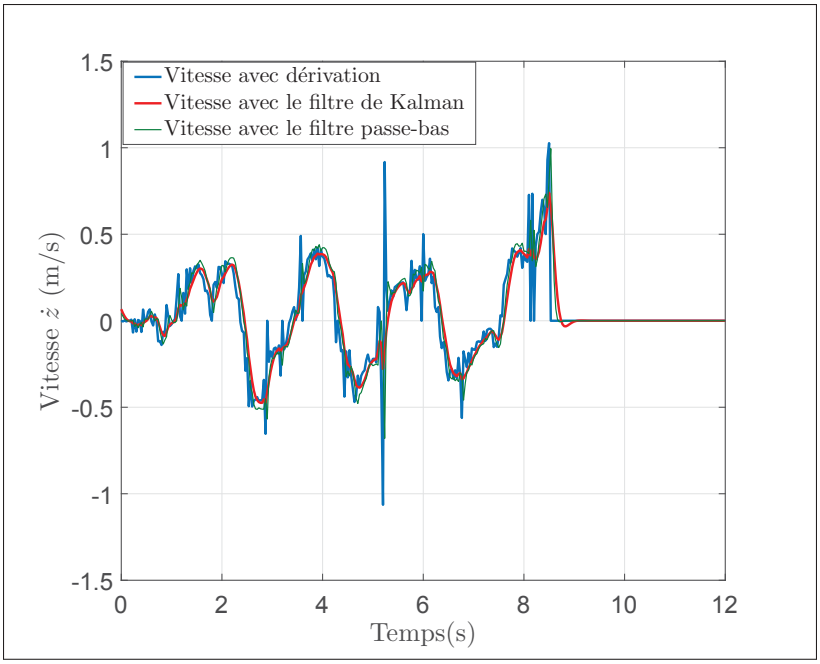

Figure 5.16 Estimation de la vitesse avec le filtre de Kalman

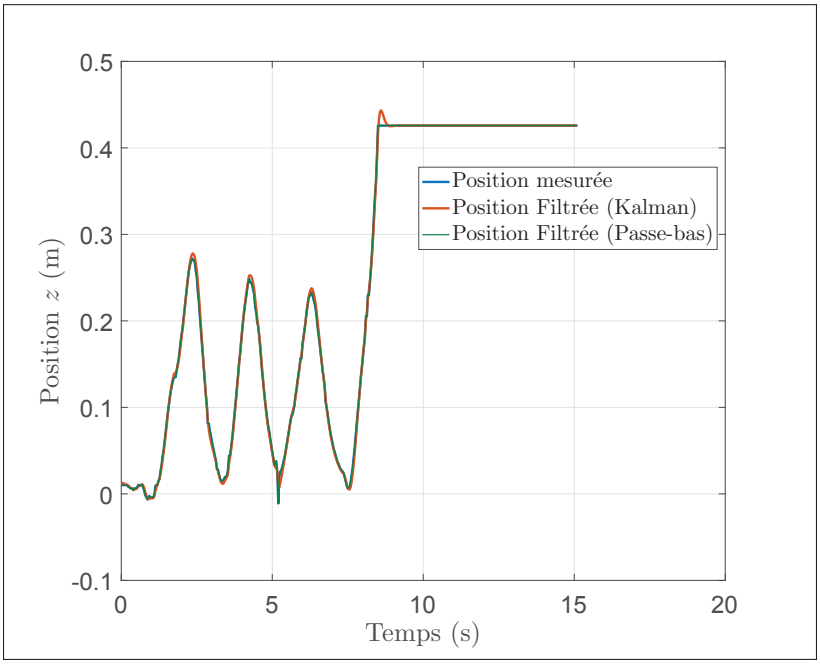

Figure 5.17 Position z_i filtrée avec le filtre de Kalman

On a vu qu'on ne pouvait pas directement dériver la vitesse à partir de la position sous risque d'avoir un résultat bruité. Deux solutions ont été présentées. Le filtre passe-bas de second ordre permet d'estimer la vitesse pour une sortie de signal faiblement bruité. Le filtre de Kalman permet d'améliorer les résultats en se basant sur un modèle où les covariances de bruit des mesures sont présentes. On a, à présent, les positions et vitesses actuelles du drone à partir de la Kinect. On a aussi récupéré les moments d'inertie des drones. On peut à présent simuler et expérimenter la commande sur un drone.

CHAPITRE 6

PLATFORME D'EXPÉRIMENTATION ET VALIDATION DE LA COMMANDE

On a vu dans les chapitres précédents les manières de modéliser un drone. À partir de là, un contrôleur a été conçu. Des méthodes différentes permettent de récupérer les paramètres du drone. On peut à présent vérifier les performances du contrôleur. On commencera par faire une simulation du modèle pour vérifier la stabilité du système. Deux vérifications expérimentales seront faites. La première expérimentation traitera la commande sur le micro drone Crazyflie. La seconde traitera sur le drone S500.

6.1 Simulation

On décide d'utiliser les paramètres du drone S500 pour la simulation. On utilisera le logiciel Matlab (MathWorks (2017)) pour la modélisation. On prendra $k_{\Theta p} = 140{,}6636$, $k_{\Theta d} = 23{,}7203$, $k_p = 2{,}5007$, $k_d = 3{,}1627$, $\Gamma_x = \Gamma_y = \Gamma_z = 5$. On prendra les valeurs des moments d'inertie mesurées à partir de la méthode expérimentale.

Les gains $k_{\Theta p}$, $k_{\Theta d}$, k_p et k_d sont calculés à partir d'un placement de pôles. Les gains du contrôleur d'attitude sont calculés de façon que la commande des angles soit plus rapide que la commande de position. On donnera pour trajectoire un cercle d'un rayon de 40 cm une période de 40 secondes et une altitude d'un mètre. Les vitesses désirées des angles ainsi que les accélérations désirées seront nulles. En effet la position des angles désirés dépend des équations (4.27) et (4.23). Il peut être difficile de relever la dérivée de ces relations. Le taux de rafraichissement du contrôleur d'attitude est cependant rapide. On peut alors prendre des vitesses et des accélérations désirées nulles. On prendra un angle de lacet nul. On simulera dans un premier temps des perturbations externes Δ_x, Δ_y et Δ_z nulles. On obtient alors les résultats suivants (Figures 6.1 et Figure 6.2) :

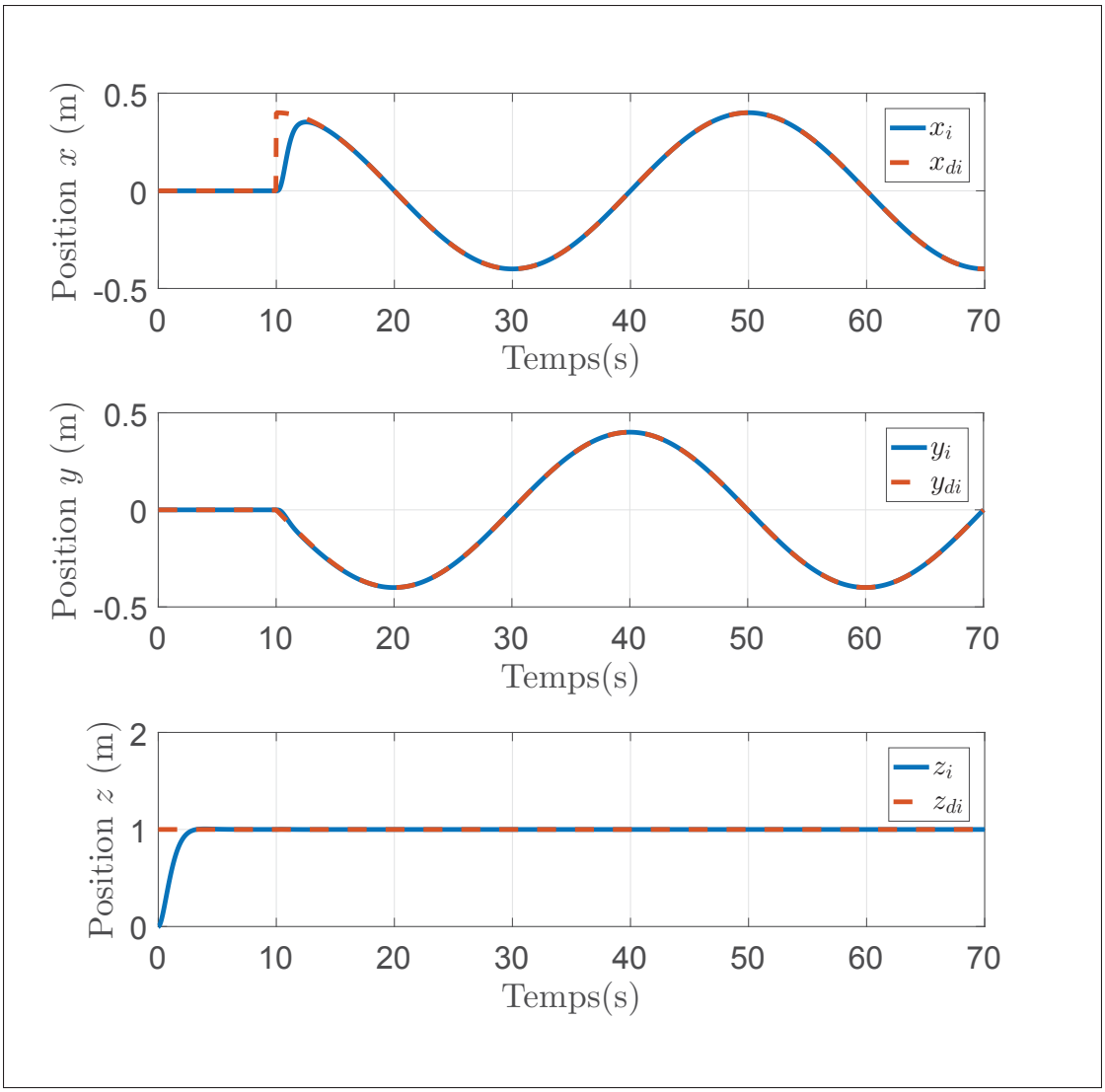

Figure 6.1 Simulation des positions

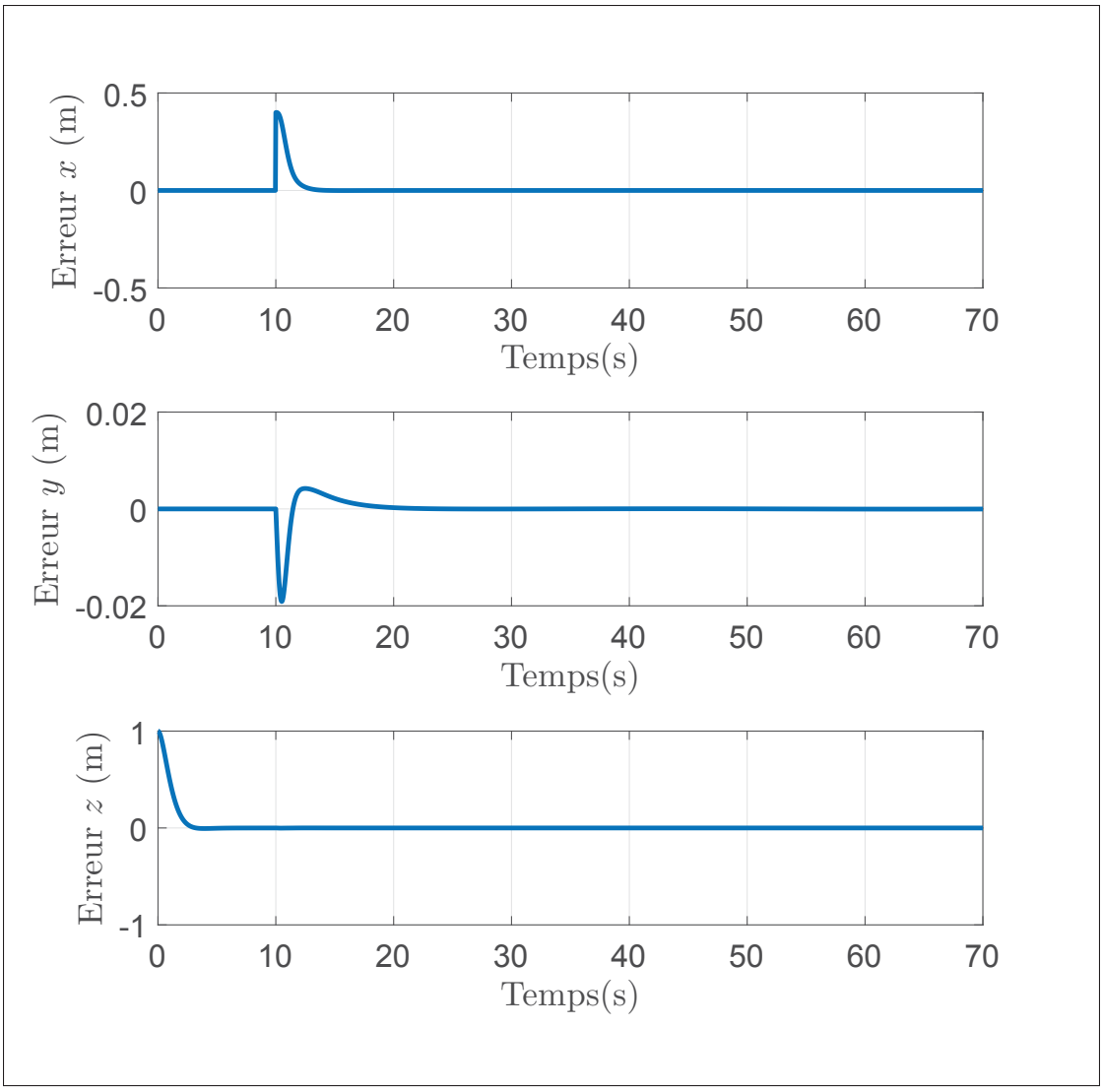

Figure 6.2 Simulation des erreurs de position

On peut voir que le temps de réponse à 5 % est de trois secondes. Ceci est dû au placement de pôles effectué grâce aux gains calculés. Ceci permettra d'avoir des couples et une force de portance faibles pour être utilisés sur les drones. Le contrôleur d'attitude donne les résultats suivants (Figure 6.3 et Figure 6.4) :

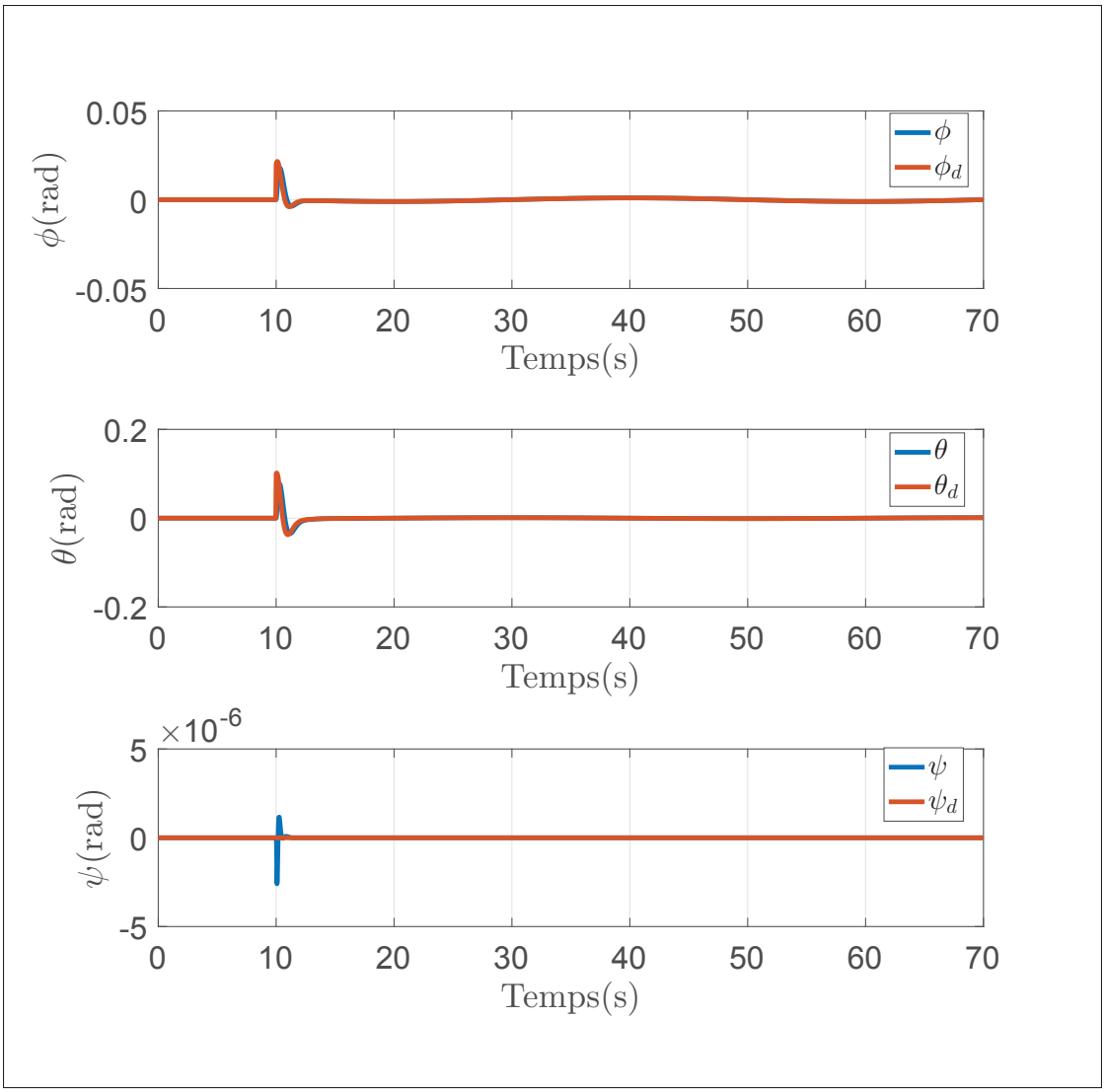

Figure 6.3 Simulation des angles

On peut voir que les angles suivent correctement la trajectoire désirée bien que les vitesses et les accélérations désirées sont nulles. Le temps de réponse est en effet beaucoup plus rapide que celui du contrôleur de position. On regarde à présent les couples et les forces de portance.

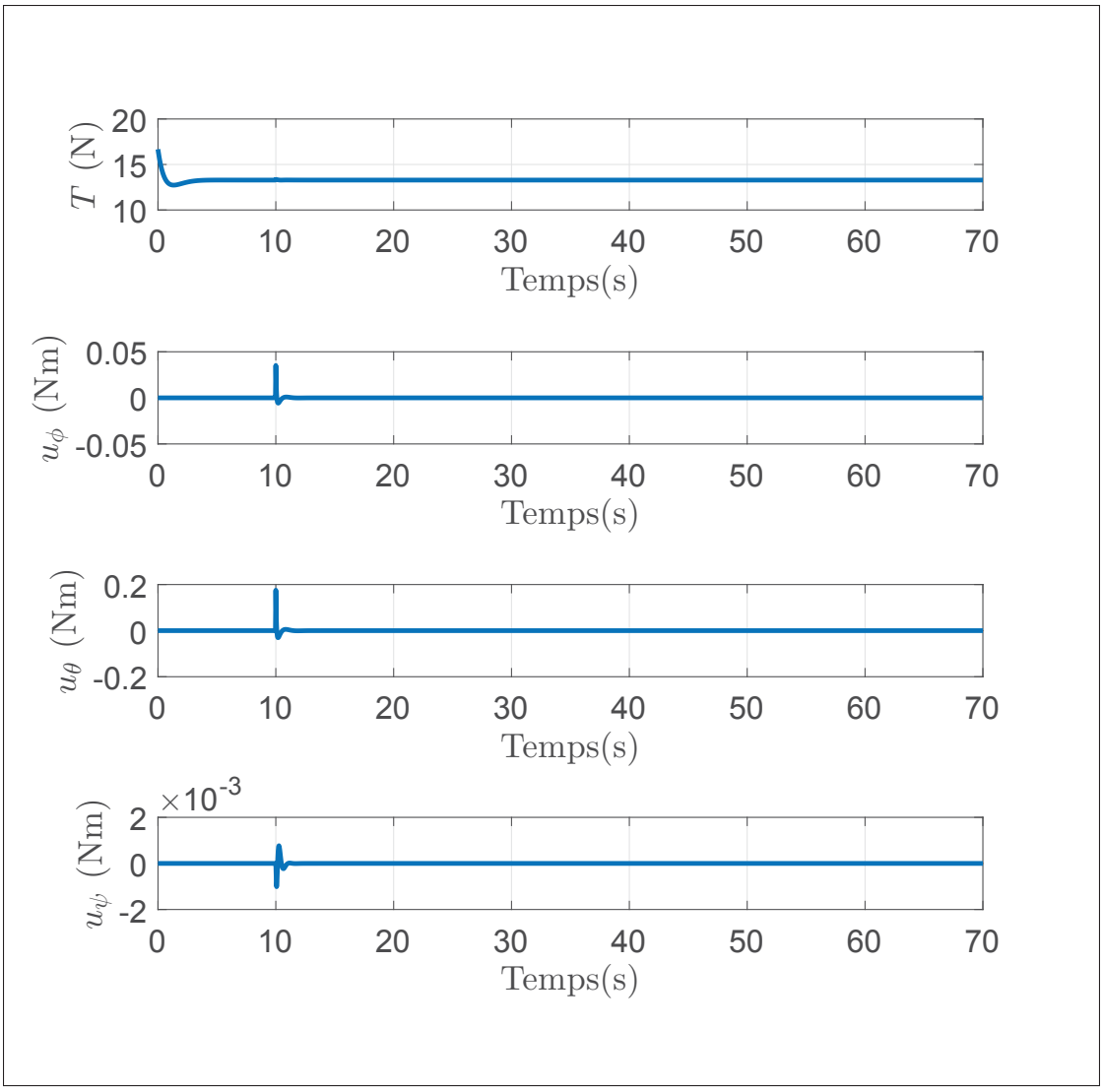

Figure 6.4 Simulation de l'effort de commande

On peut voir que l'effort de commande est conforme aux couples et à la force relevée dans le chapitre précédent. On pourra donc utiliser ce contrôleur pour les drones. On décide d'ajouter des perturbations externes telles que $\Delta_x = \Delta_y = \Delta_z = 1\ \mathrm{m} \cdot \mathrm{s}^{-2}$. On veut simuler dans un premier temps le contrôleur sans la loi d'adaptation. On obtient alors (Figure 6.5) :

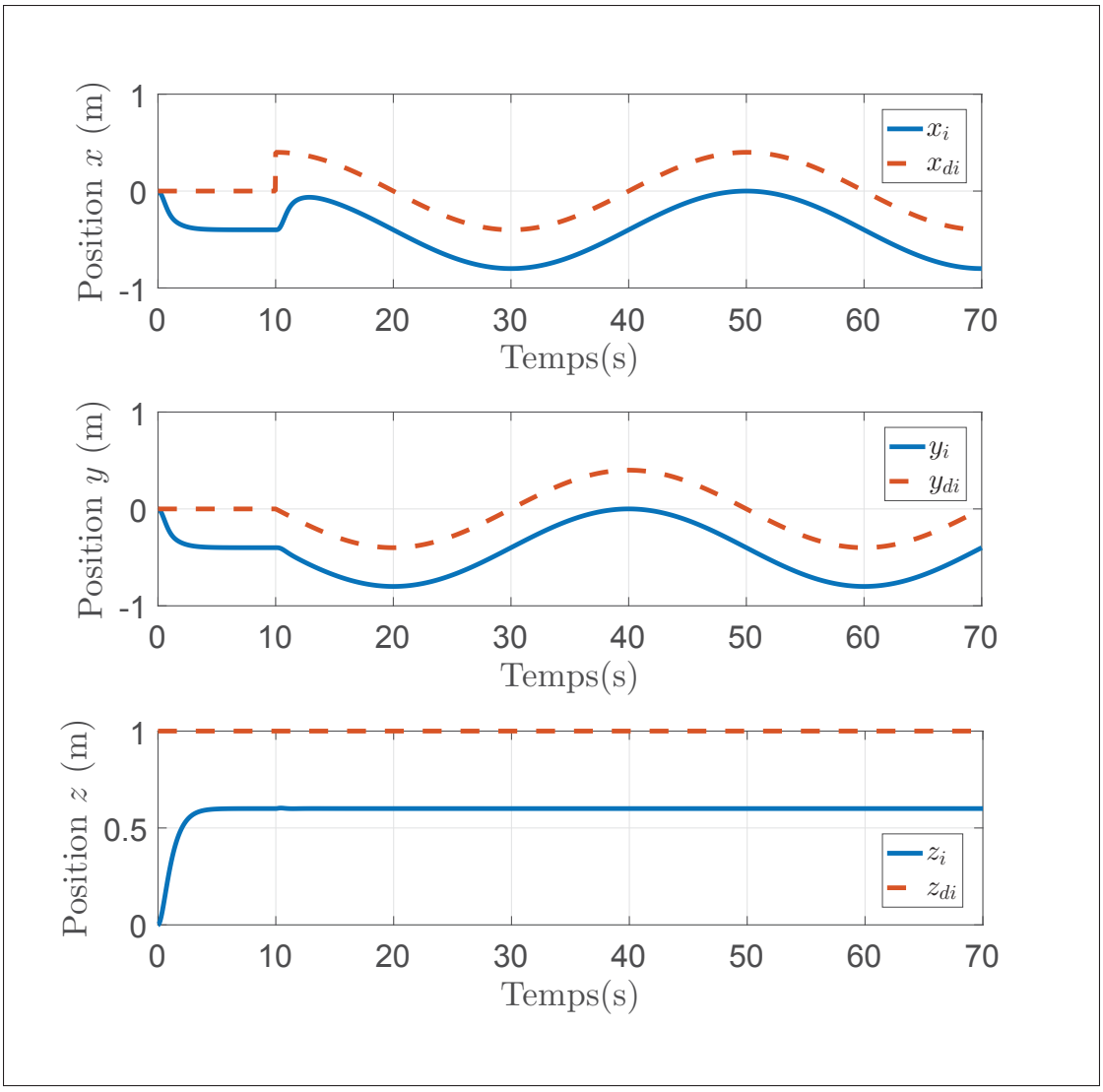

Figure 6.5 Simulation des positions avec perturbations

On peut clairement voir les erreurs de position engendrées par les perturbations. La chute de la tension de la batterie du drone peut causer ce genre de perturbation si aucun asservissement sur la tension n'a été fait. La loi d'adaptation devient alors nécessaire pour corriger ces problèmes. On obtient les résultants suivants lorsque la loi d'adaptation est utilisée (Figure 6.6) :

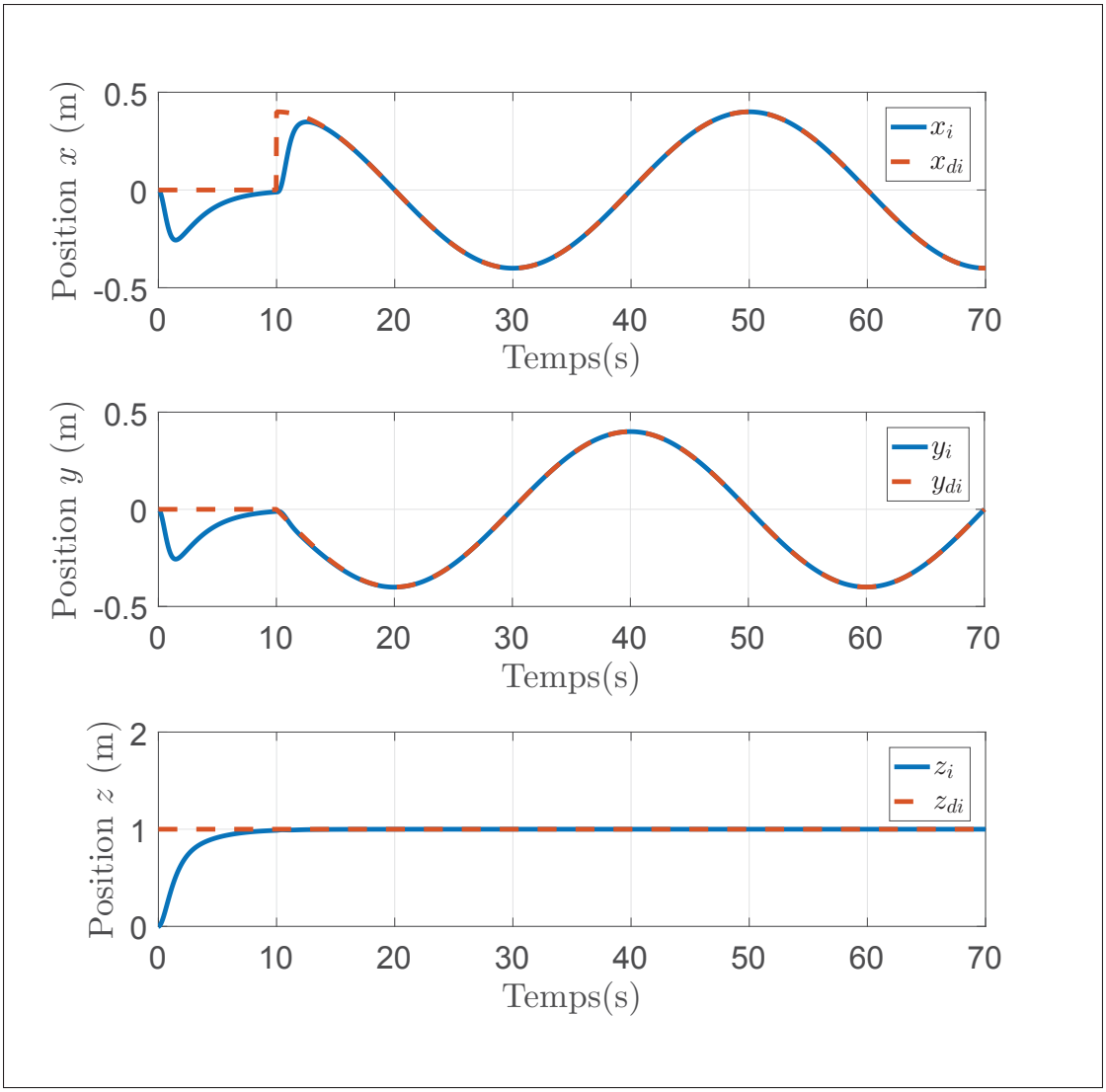

Figure 6.6 Simulation des positions avec perturbations et loi d'adaptation

On peut voir que la loi d'adaptation a permis de corriger les erreurs créées par les perturbations externes. La rapidité de la correction peut être régulée à partir des gains Γ_x, Γ_y et Γ_z. Un gain trop élevé peut cependant causer des oscillations et causer une instabilité en fonction de la nature des perturbations.

On a appliqué le contrôleur en simulation. On a remarqué que les erreurs des positions tendaient à zéro. Les couples et la force de portance appliqués sont admissibles pour le drone. On peut implémenter le contrôleur. La Figure 6.7 montre la position du drone en 3D.

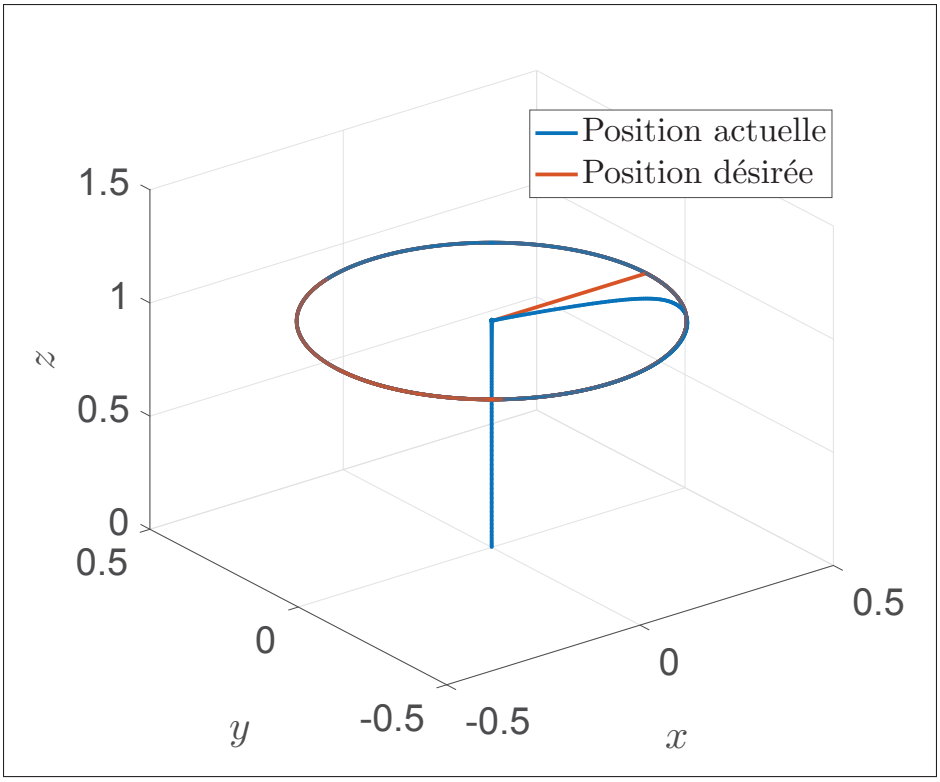

Figure 6.7 Vue 3d de la position du drone sur la simulation

6.2 Implémentation

On souhaite implémenter le contrôleur sur deux drones différents. Le premier sera le Crazyflie v2.0 et le second sera un drone où sa base est le châssis S500. Cette section évoquera les différentes manières d'implémenter et de communiquer avec ces drones. On relèvera par la suite les résultats des expérimentations. Plusieurs éléments sont nécessaires au bon fonctionnement de l'expérimentation. Une base de contrôle est nécessaire afin d'envoyer et de recevoir des informations au drone. On utilisera un ordinateur muni du système d'exploitation Ubuntu 14.04 et d'un processeur Intel Xeon E3-1200 v3. Cet ordinateur récupéra les informations de la Kinect pour exécuter l'algorithme 5.3. Les positions seront ensuite filtrées pour récupérer les vitesses

à l'aide d'un filtre de Kalman. La trajectoire sera aussi produite par l'ordinateur. Le tout sera envoyé au drone. On récupérera les positions actuelles, les angles actuels et désirés ainsi que l'effort de commande venant du drone.

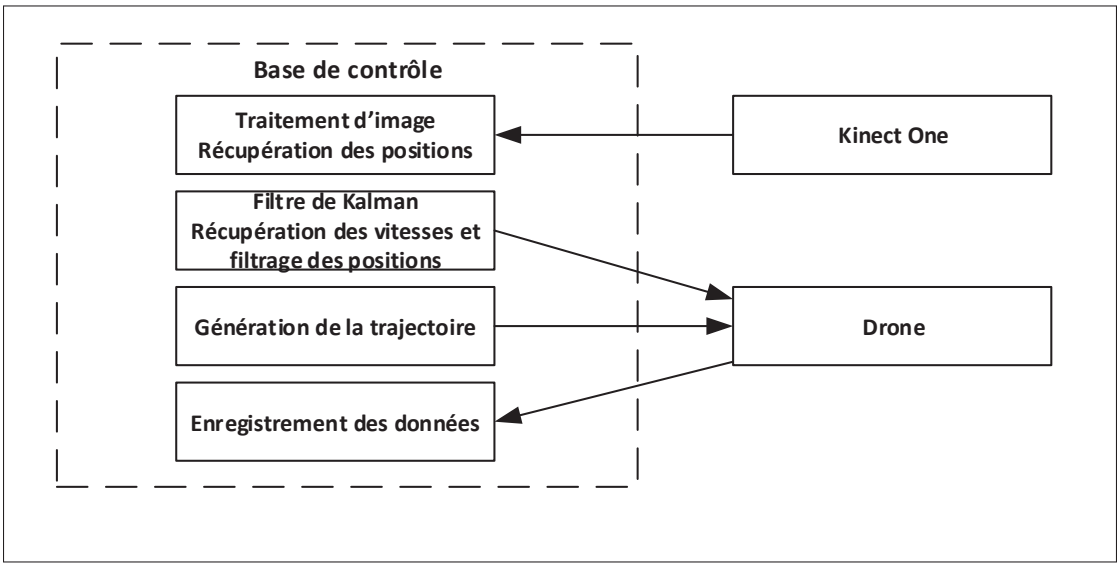

Figure 6.8 Communication de l'ensemble des périphériques

Comme on peut le voir dans la Figure (6.8), la communication est très importante. Il peut s'avérer difficile de communiquer différents appareils entre eux. Les protocoles de communication ne sont pas forcément communs. Une connaissance de ces protocoles doit être requise pour fonctionner l'ensemble du système. La programmation de la communication peut s'avérer laborieuse. Plusieurs tâches doivent être gérées simultanément et doivent être optimisées afin de recevoir ou envoyer rapidement l'information. Une solution à ce problème est d'utiliser une plateforme externe permettant de gérer les périphériques à bas niveau. On utilisera pour cela la plateforme ROS.

6.2.1 Robot Operating System (ROS)

La plateforme ROS permet de traiter les communications entre les différents appareils à bas niveau. ROS fait donc l'intermédiaire entre le protocole de communication d'un appareil et la réception et la transmission d'un message. L'utilisateur doit alors seulement savoir manipuler

ROS sans pour autant connaître aucune information sur la manière de construire le protocole et sur les adressages. La plateforme ROS se base sur un serveur (pouvant être en réseau ou local) ROS (appelé *roscore*) où tous les messages se basant sur le protocole ROS sont reçus. Ces messages peuvent être alors modifiés (on dit alors qu'on *publie* une information à un message) ou reçus (on dit qu'on *souscrit* à un message) par n'importe quel appareil étant connecté au serveur ROS. La procédure permettant de faire le point entre le protocole de communication et ROS est appelée paquet (Figure 6.9). Le paquet est donné le plus souvent par le développeur de l'appareil en question. On peut cependant créer notre propre paquet pour n'importe quel appareil de peu qu'on connaisse la structure du protocole de communication.

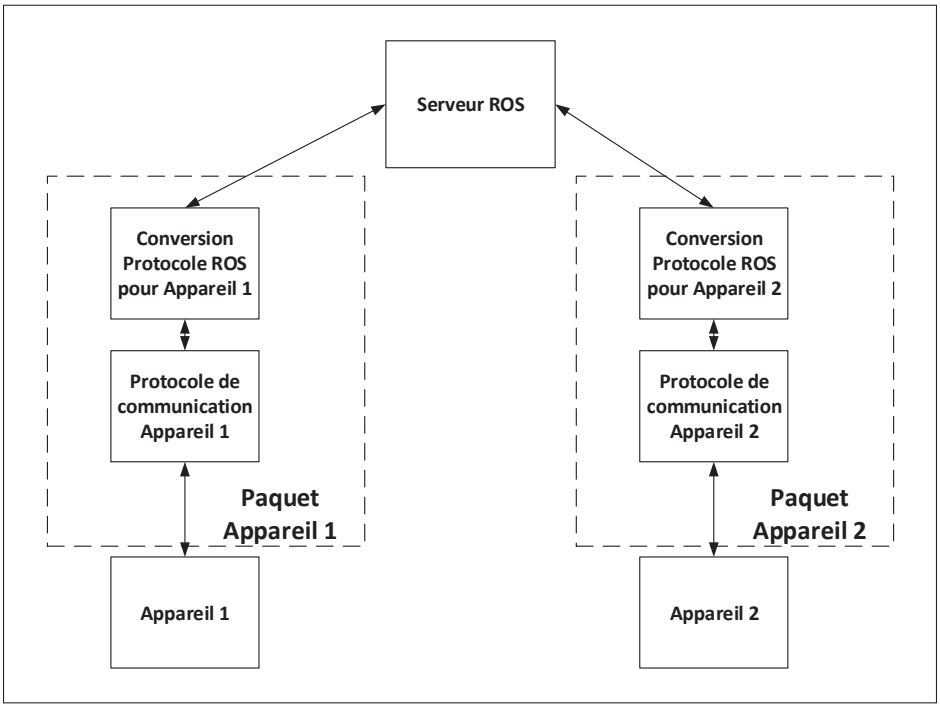

Figure 6.9 Fonction de base de ROS

Les langages de programmation compatibles ROS sont Python, C++ et Lisp. ROS contient une panoplie de librairies permettant d'envoyer tout type de structure d'un message (messages pour les positions, trajectoires, repères cartésiens, lecture d'image, lecture de centres inertiels...). Il est alors inutile de connaître la structure du protocole de ROS en bas niveau.

On utilisera plusieurs paquets pour l'implémentation du contrôleur :

a. Le paquet Iai Kinect2 permet de faire le lien entre le protocole de communication de la Kinect One et ROS (voir https://github.com/code-iai/iai_kinect2). On pourra alors récupérer les images provenant des capteurs infrarouges et de la caméra. On récupérera aussi les paramètres intrinsèques de la caméra ;
b. Le paquet Mavros (disponible sur http://wiki.ros.org/mavros) permet de faire la relation entre le protocole de communication de PX4 appelé Mavlink (une documentation est disponible sur http://qgroundcontrol.org/mavlink/start) et ROS ;
c. Crazyflie ROS (Hoenig *et al.* (2015)) permettant de faire la relation entre le protocole de communication du Crazyflie et ROS.

On créera un paquet permettant d'exécuter l'algorithme 5.3 et de créer des messages ROS contenant la position, la vitesse ainsi que la trajectoire désirée. On parlera dans les sections suivantes, de manière spécifique, l'implémentation de la commande sur le Crazyflie 2.0 et sur le Pixhawk.

6.2.2 Implémentation de la commande sur le Crazyflie

La commande sera dans un premier temps implémentée sur le micro drone Crazyflie 2.0.On modifiera le micrologiciel de base du Crazyflie pour intégrer nos contrôleurs. La communication entre la base de contrôle et le drone se fera à l'aide de la clé CrazyflieRadio (une documentation peut se trouver sur https://www.bitcraze.io/2012/02/the-crazyradio-dongle). Cette clé permet d'avoir une portée jusqu'à un kilomètre. Il dispose aussi d'un taux de transmission allant de 2 mégabits par seconde. On utilisera le protocole de communication spécifique au Crazyflie CRTP (Crazy Real Time Protocol) couplé avec ROS. Des informations sur ce protocole sont disponibles sur https://wiki.bitcraze.io/projects:crazyflie:crtp.

6.2.2.1 Protocole de communication du Crazyflie

Le micrologiciel du Crazyflie (pouvant être obtenu sur https://github.com/bitcraze/) utilise le protocole CRTP. La structure d'un paquet est constituée de 32 octets. L'en-tête est composé de 8 bits. Les bits restants correspondent à la nature des données à envoyer.

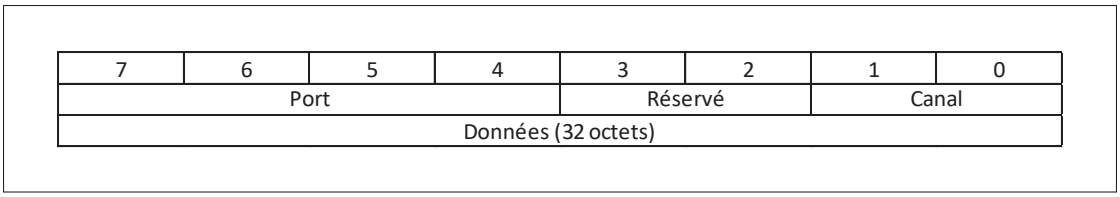

Figure 6.10 Paquet du protocole CRTP

La Figure 6.10 montre la structure du paquet. Quatre bits sont destinés au port. Le port est un nombre compris entre 0 et 15 permettant de connaître le type de message (positions, angles, commandes...). Les deux bits de canal permettent d'envoyer deux types de messages sur un même port. On utilisera que le canal 0. Le Tableau 6.1 montre la structure d'un paquet du protocole Crazyflie.

Tableau 6.1 Ports utilisés pour l'implémentation du Crazyflie

Port	**Nom**	**Description**
0	Console	Lit la console
2	Paramètres	Modifie/Lit les paramètres du drone
3	Commande	Envoie de la trajectoire pour les angles et la force de portance
5	Log	Récupère les données
6	Position externe	Envoie de la position actuelle

Le Tableau 6.1 montre les différents ports pouvant être utilisés par défaut. On utilisera le port 2 pour envoyer les valeurs des différents gains. On récupéra les angles actuels ainsi que les efforts de commandes à l'aide du port 5. On doit envoyer la trajectoire ainsi que les positions actuelles. On ne dispose cependant pas de ports suffisants pour directement envoyer les informations. Le port 6 demande en effet que trois valeurs pour la position. Le port 3 demande quatre différentes valeurs pour les angles et la force de portance (voir la référence du protocole CRTP). On décidera alors d'utiliser le port 6 non pour envoyer la position actuelle, mais pour envoyer les trois erreurs de position. On utilisera le port 3 pour envoyer les erreurs de vitesses multipliées par le gain k_d ainsi que les accélérations désirées (les accélérations désirées étant sommées aux erreurs de vitesse multipliées par k_d).

L'accès à ces ports peut être obtenu à l'aide de ROS. Le paquet Crazyflie ROS contient des messages permettant d'écrire sur les ports 3 et 6 (voir la référence du paquet Crazyflie ROS). Le paquet ROS s'occupera par la suite d'envoyer à l'aide de la clé Crazyflieradio les différents messages. La Figure 6.11 résume la communication entre la base de contrôle et le drone.

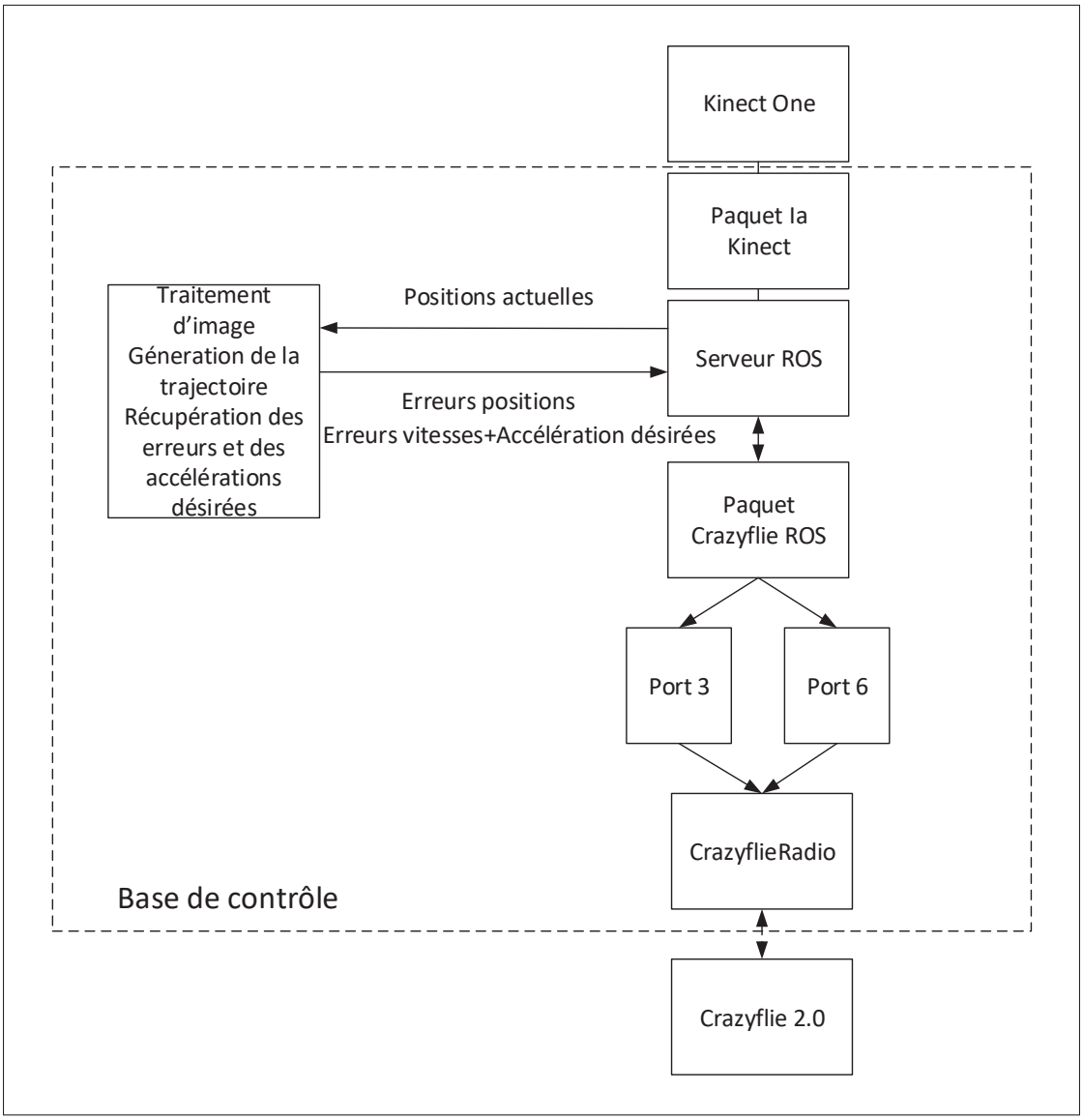

Figure 6.11 Communication entre la base de contrôle et le Crazyflie

6.2.2.2 Résultats de l'implémentation sur le Crazyflie

On décide de faire pour la trajectoire un cercle de rayon 50 cm et une période de 30 secondes. On prendra les paramètres du drones décrits par Landry (2015). On prendra de même $k_{\Theta p} = 140{,}6636$, $k_{\Theta d} = 23{,}7203$, $k_p = 2{,}5007$, $k_d = 3{,}1627$, $\Gamma_x = \Gamma_y = \Gamma_z = 0.01$. On obtient alors les résultats suivants (Figure 6.12 et Figure 6.13) :

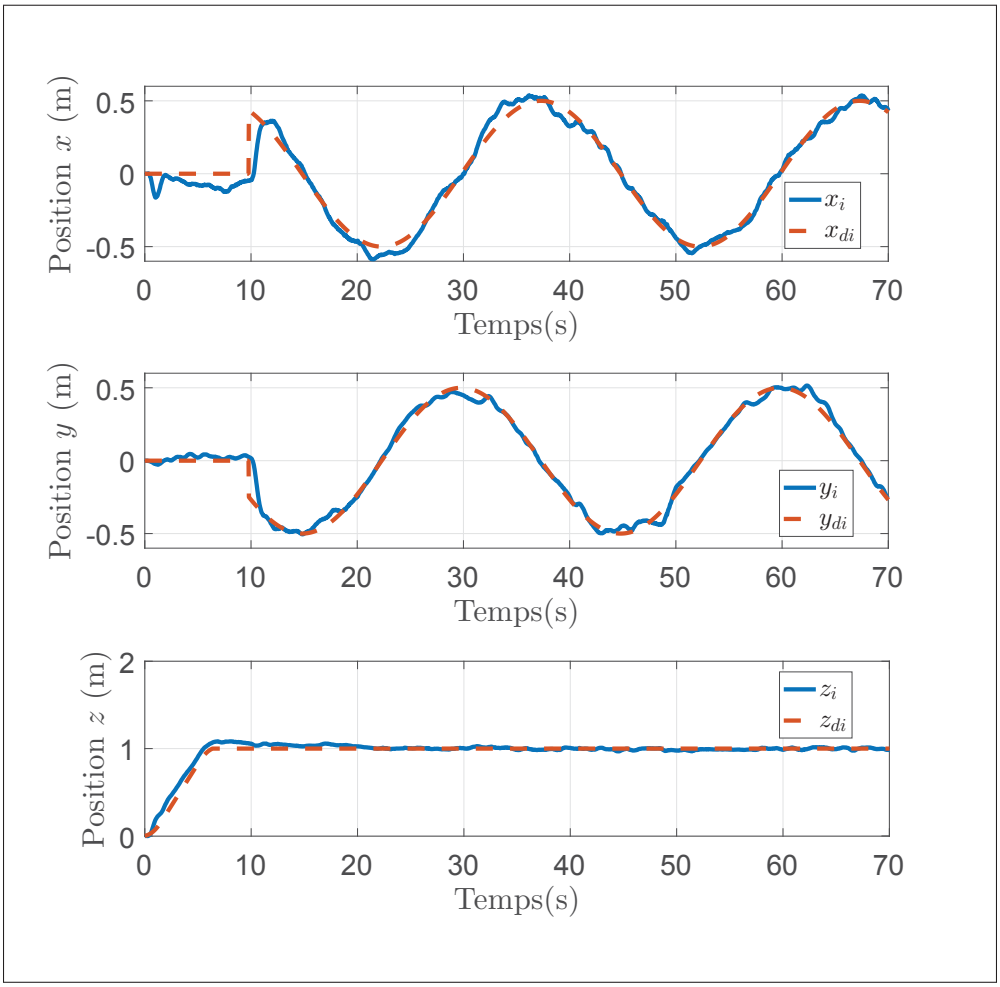

Figure 6.12 Résultats des positions avec le Crazyflie

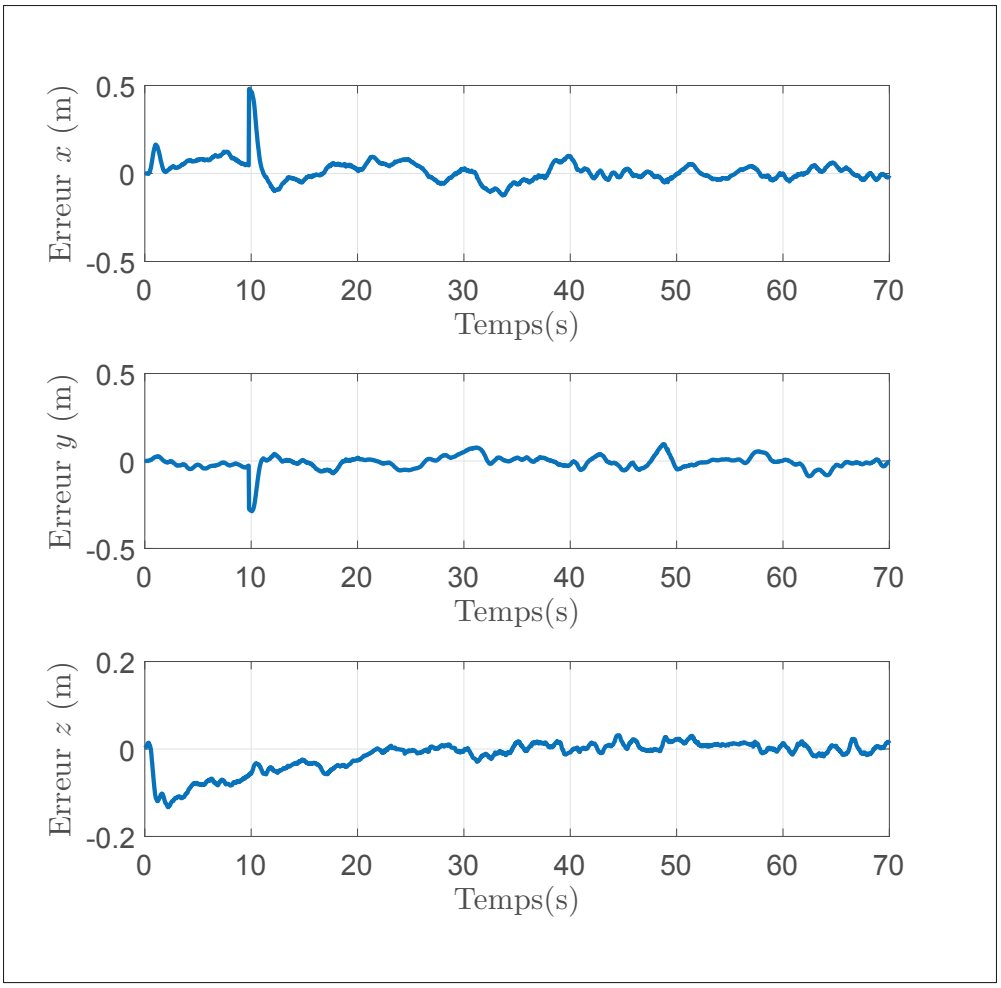

Figure 6.13 Résultats des erreurs de position avec le Crazyflie

On obtient des erreurs en dessous de dix centimètres. Le temps de réponse à 5 % est d'environ trois secondes. On peut remarquer ce temps lorsque la trajectoire passe subitement de zéro à une certaine valeur pour commencer le cercle. La Figure 6.14 permet d'avoir une vue en 3D du comportement du drone.

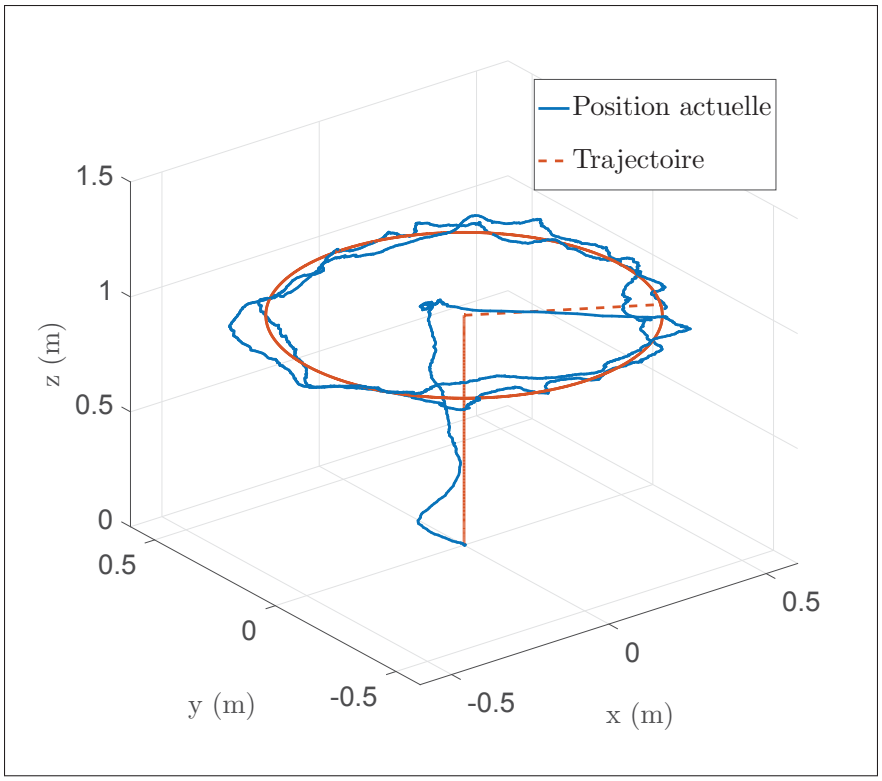

Figure 6.14 Résultats en 3D de la position avec le Crazyflie

On peut voir qu'une erreur d'environ 15 cm est obtenue lorsqu'on se trouve dans la phase du décollage. Ceci s'explique par l'action intégrale de la loi d'adaptation qui engendre un dépassement au début du lancement. Le drone se stabilise ensuite tout au long de la trajectoire pour se retrouver avec une erreur pour les positions x et y entre -10 et 10 cm. L'erreur de la position en z se situe dans un intervalle de - 5 et 5 cm en régime permanent. On souhaite à présent regarder la réponse des angles ainsi que les efforts de commande (Figure 6.15).

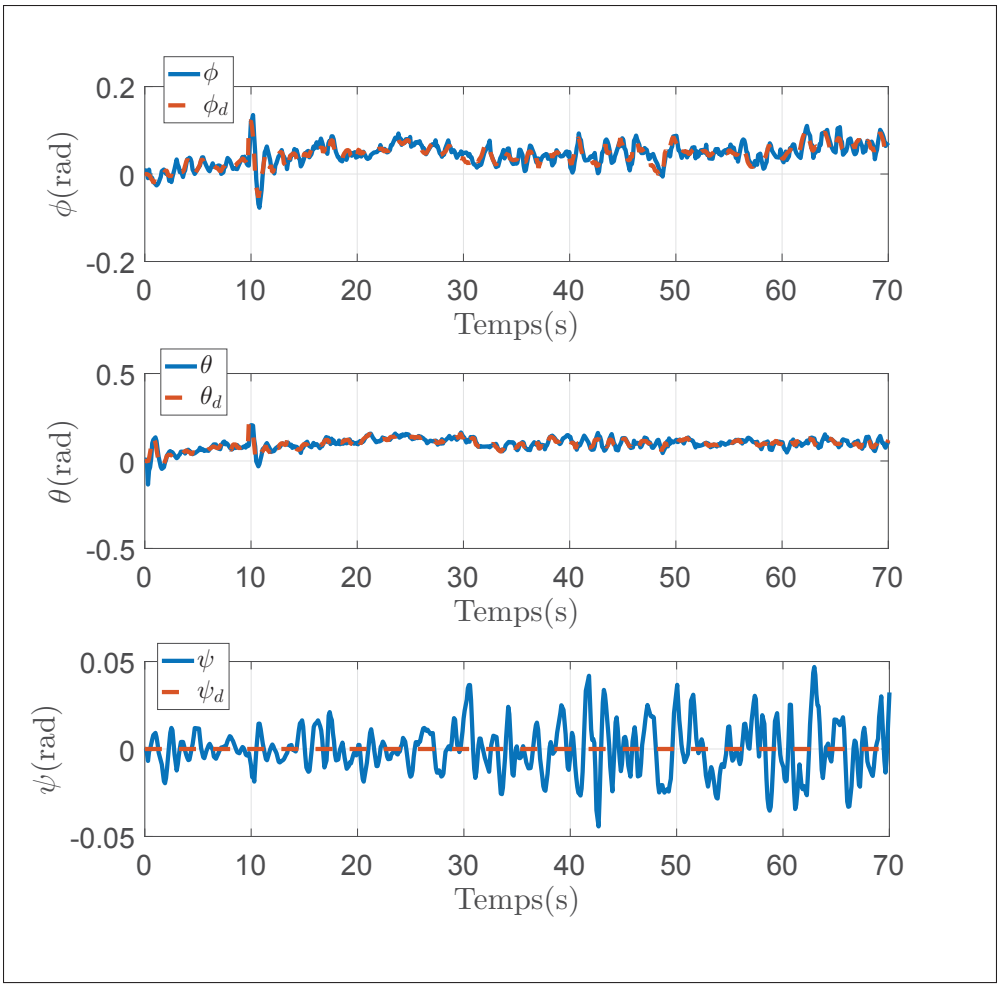

Figure 6.15 Résultats des angles avec le Crazyflie

Les angles varient entre −0,1 et 0,1 radian. Leurs variations ne sont pas agressives ce qui assure la stabilité du drone. Les erreurs des angles se situent entre −0,06 et 0,06 radian en régime permanent. On regarde à présent les efforts de commande (Figure 6.16).

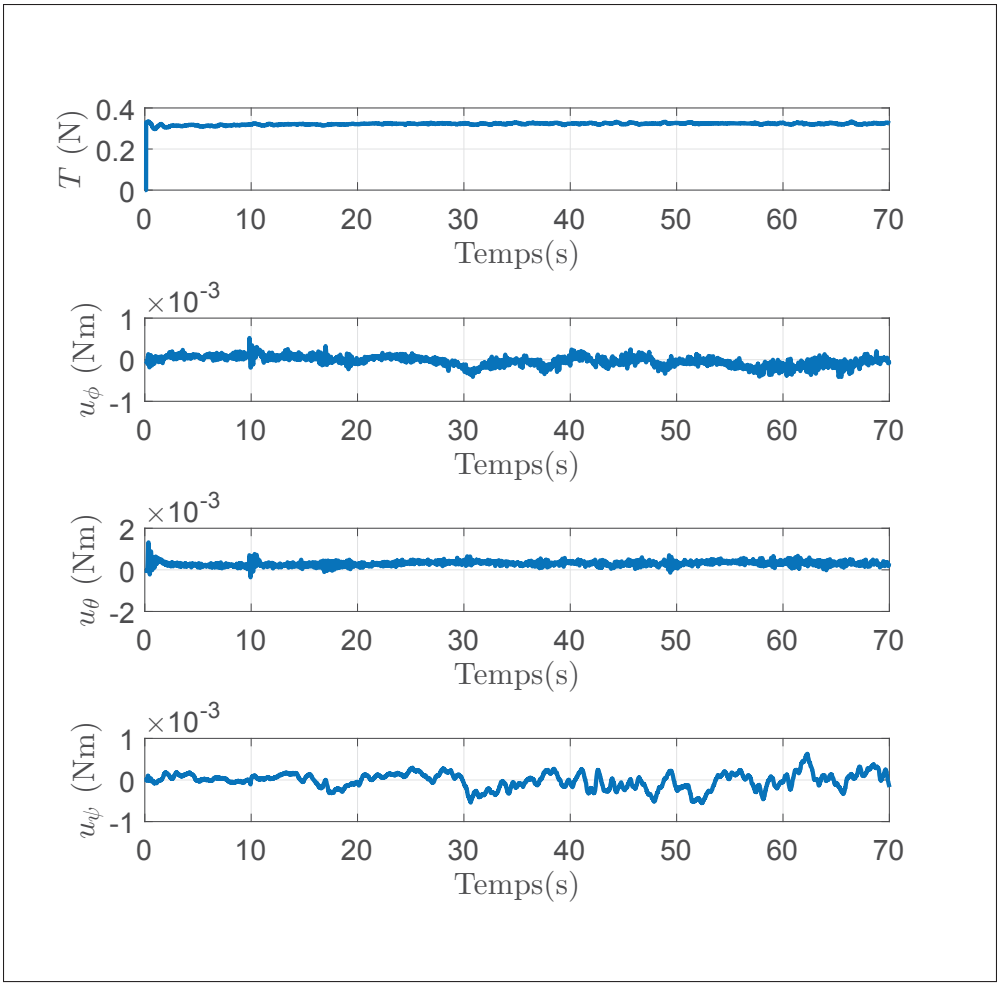

Figure 6.16 Résultats des efforts de commande avec le Crazyflie

On peut voir que la force de portance se situe au voisinage de la force de planage. Les variations des couples sont faibles ce qui assure la stabilité du système.

6.2.2.3 Conclusion

L'utilisation de ROS a permis de contrôler à partir d'une base de contrôle le drone Crazyflie. Les résultats de l'expérimentation montrent que nous obtenons un suivi de trajectoire avec une erreur se situant entre -10 et 10 cm. Cette erreur peut s'expliquer par le fait que le microdrone est très sensible aux perturbations extérieures dû à sa taille. La faible variation de l'effort de commande permet d'avoir une stabilité tout au long de l'expérimentation. Le contrôle d'un

micro drone peut donc se faire à partir de la Kinect One. On veut à présent implémenter le contrôleur sur le drone S500.

6.2.3 Implémentation de la commande sur le drone S500

On a vu l'implémentation de la commande sur un micro drone. Les applications sur un micro drone sont cependant limitées par la charge utile que peut supporter le drone. On veut alors connaître l'efficacité de la commande avec la Kinect One sur des drones étant plus grands et plus lourds. On utilisera pour cela le drone S500 muni d'un Pixhawk et d'un ordinateur embarqué Odroid XU4. On utilisera un routeur WI-FI afin de communiquer avec le drone à partir d'une base de contrôle. Le drone utilise un protocole de communication appelé Mavlink. L'utilisation de ROS permettra de faire le lien entre ce protocole et la base de contrôle.

6.2.3.1 Protocole de communication Mavlink

Contrairement au protocole CRTP, Mavlink est un protocole utilisé par une panoplie de micrologiciel. Mavlink propose une structure de protocole permettant de contrôler différents types de drones. Beaucoup de drones dans l'industrie utilisent ce protocole. Ceci permet aussi aux différents micrologiciels utilisant Mavlink d'être compatible avec la plupart des stations terrestres. Un paquet de Mavlink peut aller de 8 à 263 octets. On peut envoyer une donnée faisant 256 octets au maximum. Le Tableau 6.2 montre la structure d'un paquet du protocole Mavlink.

Tableau 6.2 Structure d'un paquet Mavlink

Numéro de l'octet	**Nom**
0	Commencement du paquet
1	Taille de la donnée
2	Numéro de la séquence d'un paquet
3	ID du système receveur
4	ID du composant
5	ID du message
6 à (n+6)	Données
(n+7) à (n+8)	Octets de contrôle

Une documentation (Mavlink (2015)) permet d'avoir la liste des différents messages MAVLINK disponibles. On utilisera pour notre cas les messages décrits dans le Tableau 6.3.

Tableau 6.3 Messages Mavlink utilisés

Message	**Description**
VISION POSITION ESTIMATE	Envoie la position actuelle
VISION SPEED ESTIMATE	Envoie la vitesse
SET POSITION TARGET LOCAL NED	Envoie la position, vitesse et accélération désirées
ATTITUDE	Récupère les angles actuels

On utilisera, comme avec le Crazyflie, ROS pour établir le pont entre Mavlink et le drone. Le Pixhawk ne dispose pas d'antenne WI-FI. On utilisera l'ordinateur embarqué Odroid XU4 sur le drone afin d'établir une communication entre le drone et la base de contrôle. On communiquera avec l'Odroid à l'aide de ROS où le serveur sera sur un réseau (Figure 6.17).

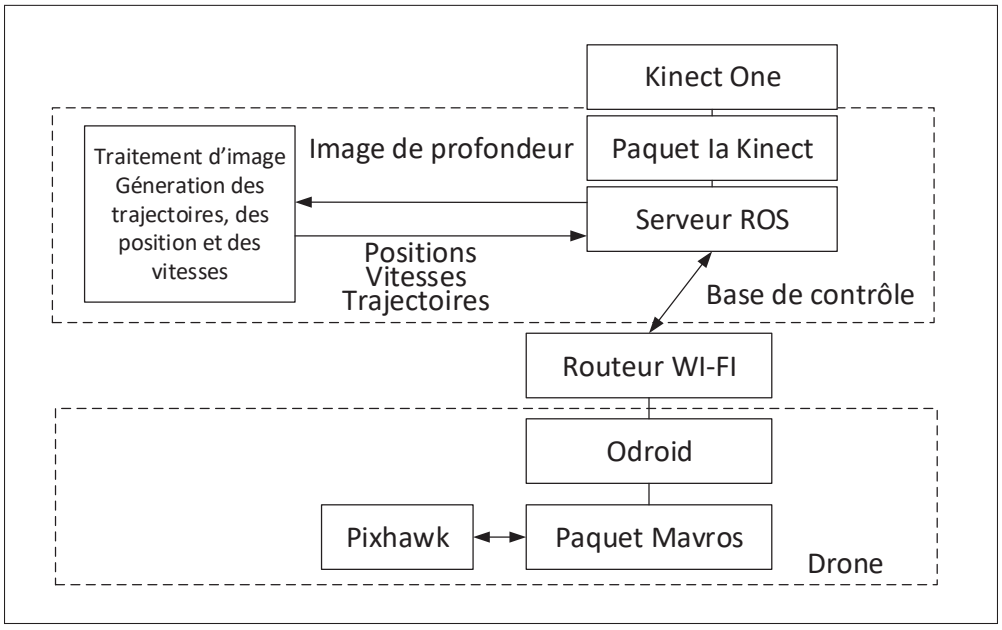

Figure 6.17 Communication entre la base de contrôle et le drone S500

6.2.3.2 Résultats expérimentaux du drone S500

On décide de faire pour la trajectoire un cercle de 40 cm d'une période de 40 secondes. On prendra $k_{\Theta p} = 140{,}6636$, $k_{\Theta d} = 23{,}7203$, $k_p = 1{,}4066$, $k_d = 2{,}3720$, $\Gamma_x = \Gamma_y = \Gamma_z = 0{,}7$. On prendra les valeurs des moments d'inertie du drone S500 mesurées à partir de la méthode expérimentale. La Figure 6.18 montre le suivi de trajectoire du drone, la Figure 6.19 présente les erreurs de position.

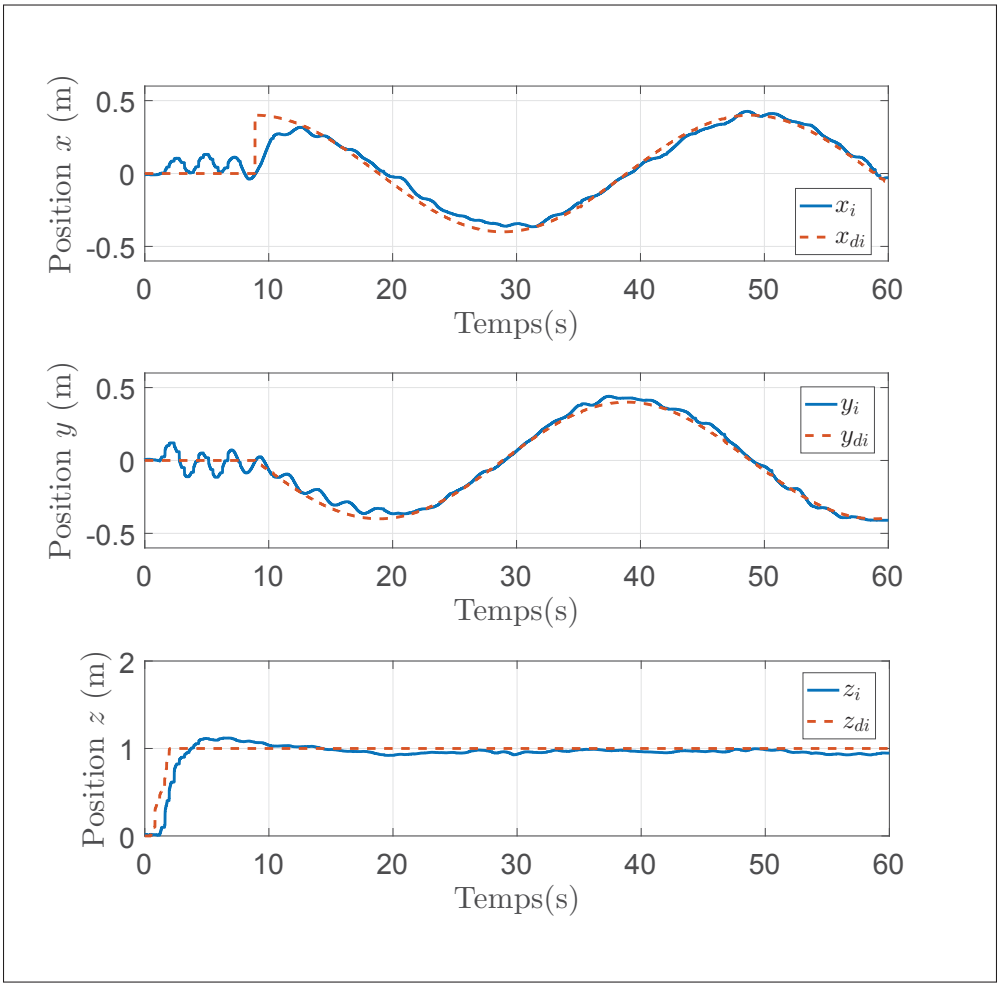

Figure 6.18 Résultats des positions avec le drone S500

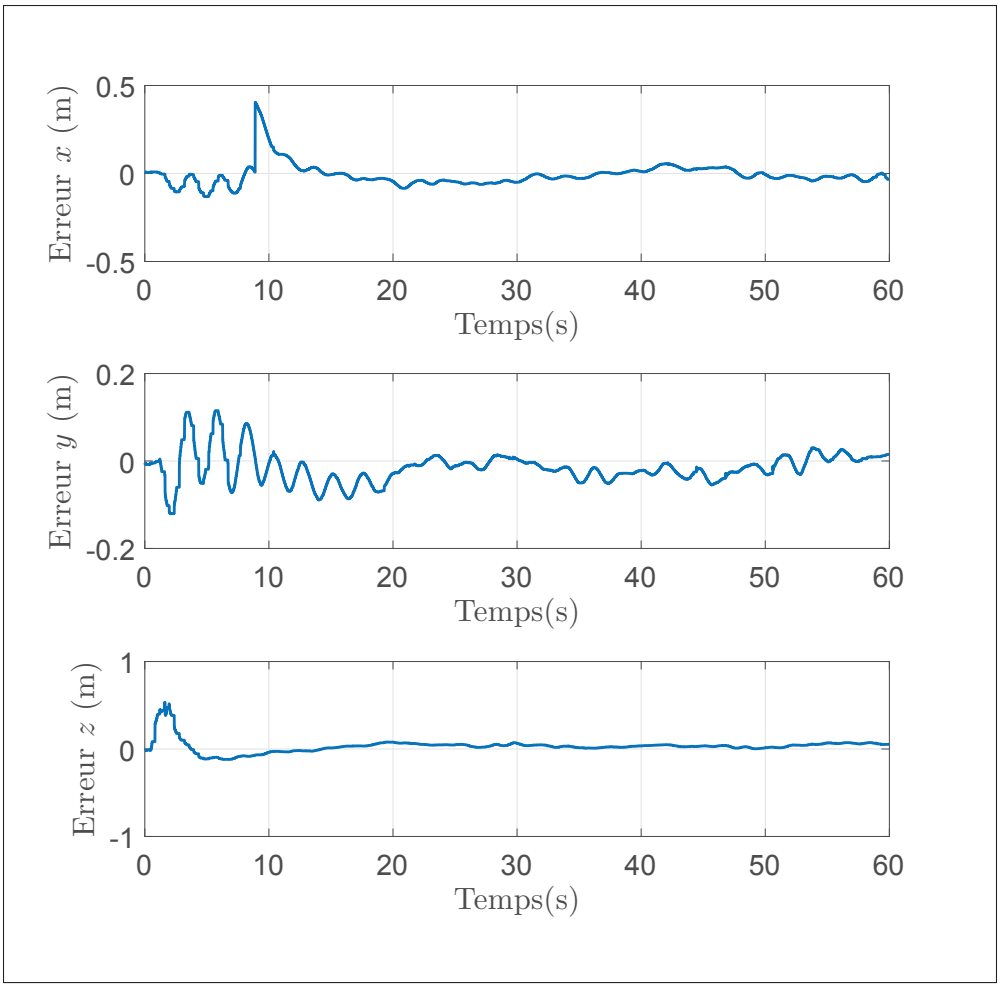

Figure 6.19 Résultats des erreurs de position avec le drone S500

On peut voir que nous avons au début de la trajectoire une oscillation d'environ 10 centimètres pour x et pour y. Ceci est dû à la loi d'adaptation et à la chute de la tension au démarrage des moteurs. Une réduction des gains Γ_x, Γ_y et Γ_z permet de résoudre le problème. Les erreurs de position du drone prendront cependant plus de temps à converger. On peut cependant remarquer qu'à partir de dix secondes les erreurs se situent entre -5 et 5 cm. La Figure 6.20 permet de regarder la trajectoire en 3D.

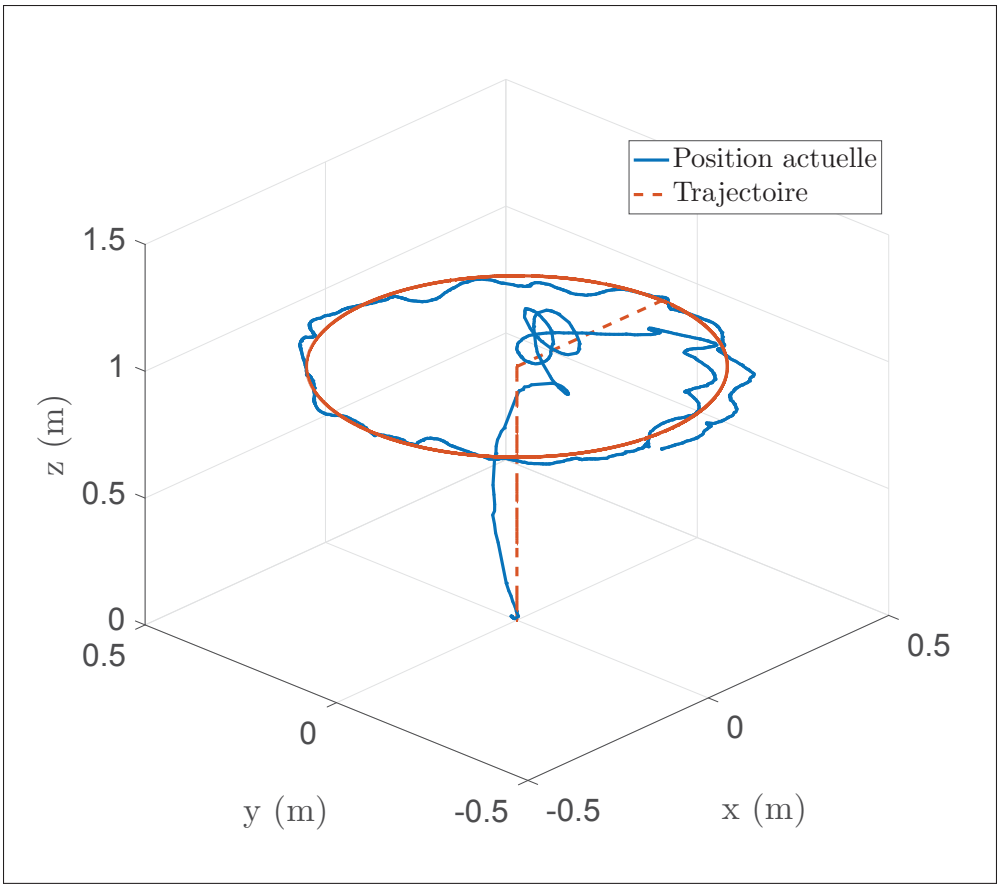

Figure 6.20 Résultats de la position du drone S500 en 3D

On peut voir l'oscillation au début lorsque le drone essaie d'atteindre 1 mètre. Le drone commence alors à se stabiliser pour suivre la trajectoire. On veut regarder le comportant des angles ainsi que les efforts de commande. La Figure 6.21 montre le suivi de trajectoire des angles d'orientation.

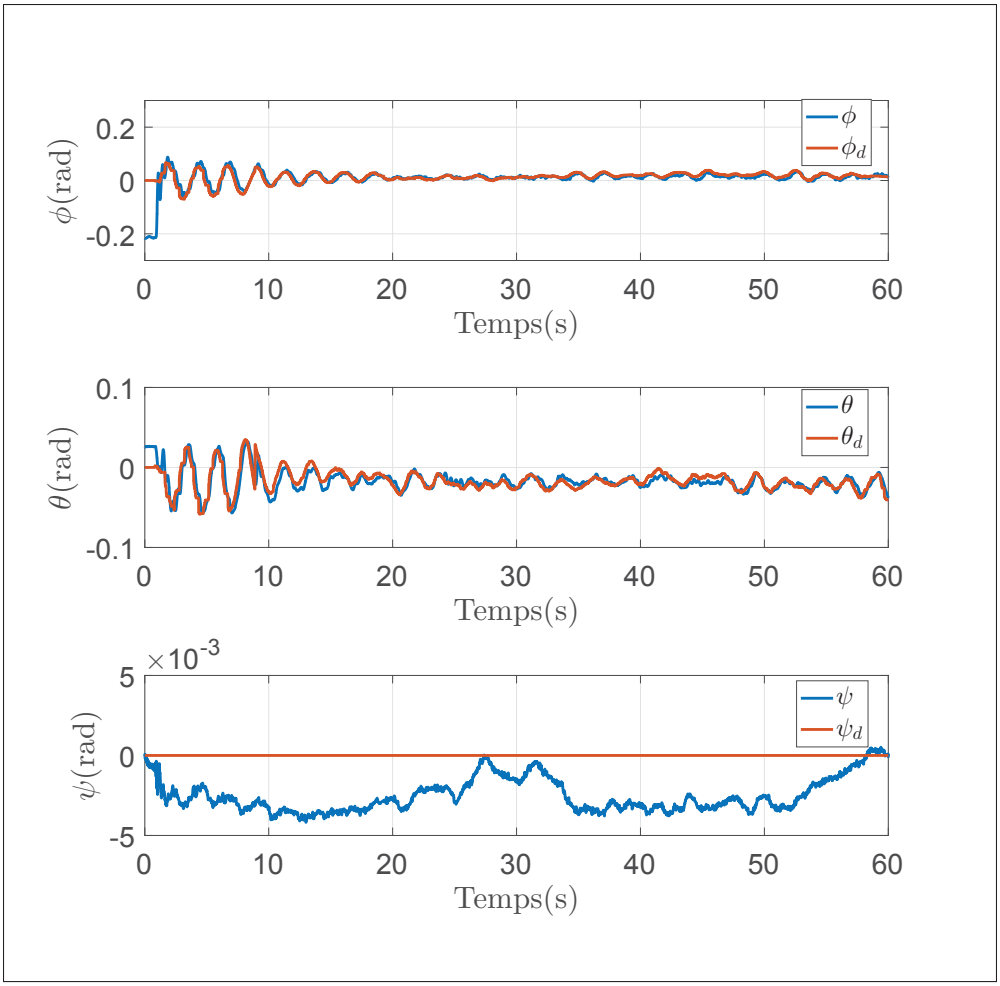

Figure 6.21 Résultats des angles avec le drone S500

Les erreurs des angles se trouvent entre −1,5 et 1,5 degré. L'angle de roulis dispose d'une oscillation d'une altitude de 5 degrés pour se stabiliser en régime permanent. On regarde à présent les efforts de commande (Figure 6.22).

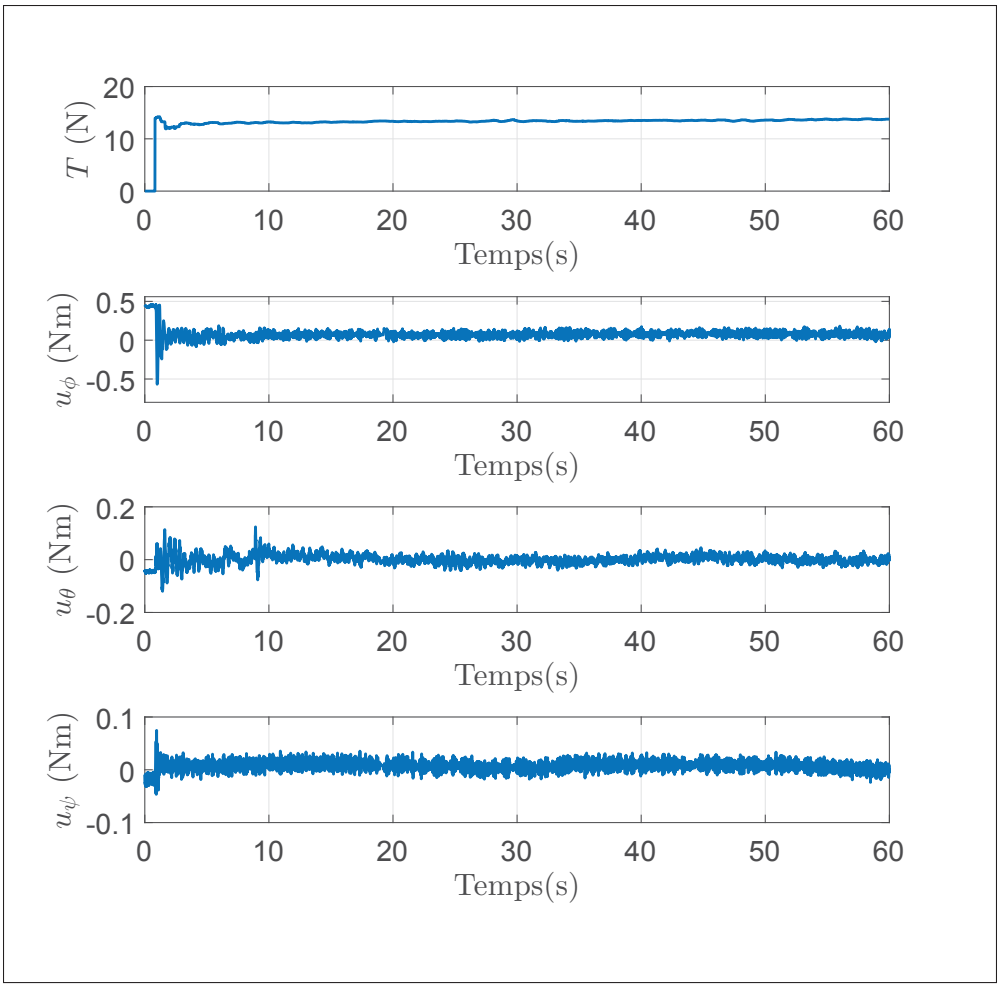

Figure 6.22 Résultats des efforts de commande avec le drone S500

On peut voir que l'effort de commande n'est pas agressif en régime permanent. On peut cependant voir la force de portance augmentée au cours du temps. Ceci est dû à la chute de tension de la batterie. Les estimations $\hat{\Delta}_x$, $\hat{\Delta}_y$ et $\hat{\Delta}_z$ aident à corriger le problème comme le montre la Figure 6.23. En effet, l'estimation $\hat{\Delta}_z$ augmente lorsque la tension de la batterie chute.

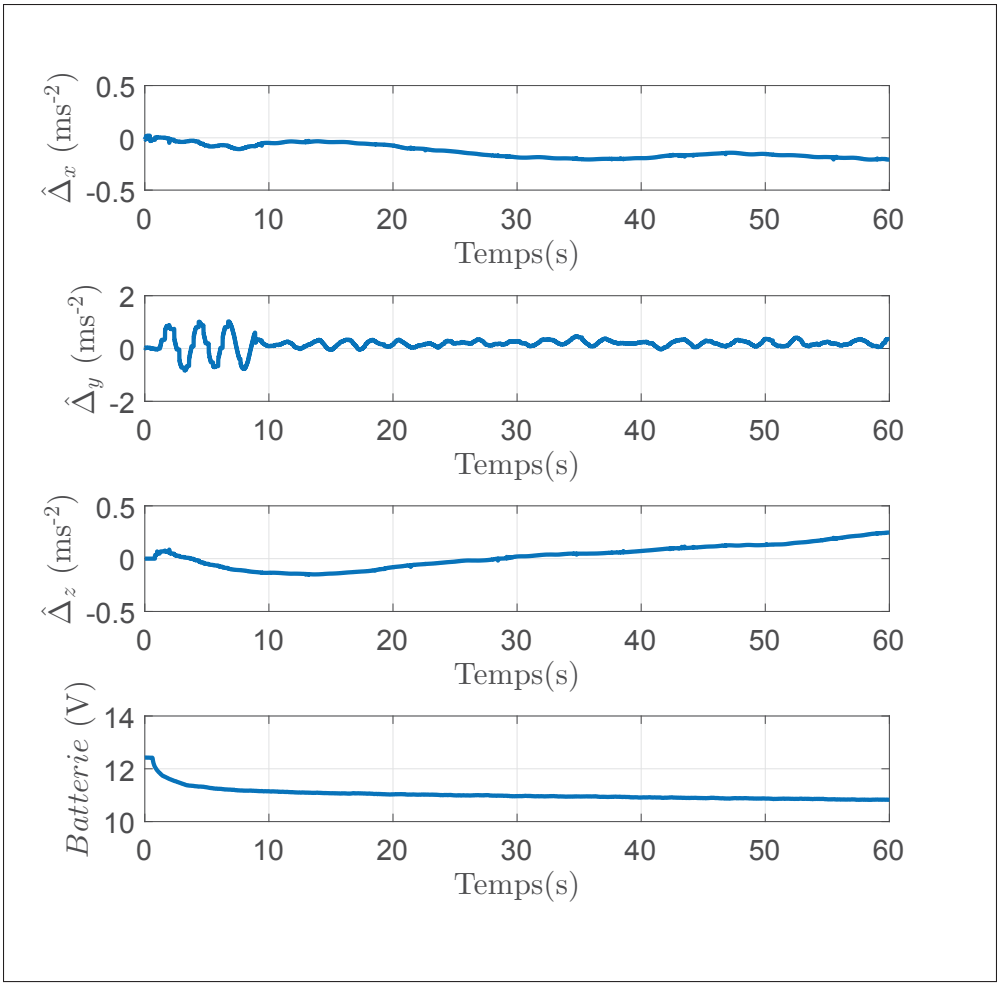

Figure 6.23 Influence des estimations des perturbations sur la chute de la tension de la batterie

6.2.3.3 Conclusion

Le contrôleur de position et d'attitude a été expérimenté à l'aide de la Kinect One, du Pixhawk et de l'Odroid. La communication entre la base de contrôle et le drone a été faite à l'aide d'un routeur WI-FI et de ROS. La commande adaptative a permis de contrer l'effet de la chute de tension de la batterie du drone. On obtient alors en régime permanent une suivie de trajectoire avec une erreur suitée entre -5 et 5 cm.

CONCLUSION

On a pu, à partir d'une modélisation du quadcopter, concevoir deux contrôleurs. Le premier, permettant de contrôler l'orientation, se base sur une linéarisation entrée-sortie. Le deuxième, contrôlant la position, se base sur une fonction de Lyapunov afin d'assurer la stabilité du système. Une loi d'adaptation a été ajoutée afin d'estimer différentes perturbations extérieures.

Les positions ont pu être récupérées grâce à la méthode de soustraction de l'arrière-plan et en utilisant la Kinect One. La dérivation directe de la position pour récupérer les vitesses crée cependant du bruit. Les vitesses ont été alors estimées à partir d'un filtre passe-bas de second ordre dans un premier temps. Ce filtre a été comparé à un filtre de Kalman. Le bruit a été correctement filtré pour les deux filtres.

Les contrôleurs de position et d'attitude demandent certains paramètres afin de pouvoir fonctionner. Les premiers paramètres requis sont la masse du drone ainsi que les moments d'inertie. Contrairement à la masse du drone, les moments d'inertie peuvent être plus compliqués à obtenir. Une méthode théorique a été alors utilisée. Cette méthode a été comparée à une autre méthode expérimentale. Les résultats obtenus sont proches les uns des autres ce qui assure la fiabilité des méthodes.

Une simulation du drone a été conçue à l'aide du logiciel Matlab pour valider les contrôleurs. La nécessité d'une loi d'adaptation a été montrée en simulant plusieurs perturbations extérieures. L'implémentation de ces contrôleurs a été faite sur deux types de drones : un micro drone Crazyflie ainsi qu'un drone S500 basé sur le Pixhawk.

L'utilisant de la caméra Kinect a permis et l'implémentation des contrôleurs sur le Pixhawk et sur le Crazyflie ont permis aux drones d'avoir une suive de trajectoire en cercle.

La Kinect One peut donc être utilisée pour implémenter des contrôleurs sous plusieurs conditions. Le drone doit être le seul objet en mouvement dans le champ de vision de la Kinect. Le

volume de l'espace de l'expérimentation ne doit dépasser pas 4 mètres cubes. Enfin les trajectoires utilisées doivent être lentes en raison du faible taux d'image par seconde de la Kinect.

RECOMMANDATIONS

La méthode de soustraction de l'arrière-plan peut avoir ses limitations. Cette méthode permet en effet de récupérer en sortie une image où seulement le drone est affiché. La détection du centre du drone peut être alors difficile à obtenir si la taille du drone est grande. Une solution à ce problème est de détecter un marqueur de petite taille au lieu du drone en entier. La méthode de soustraction de l'arrière-plan ne suffit alors pas pour isoler le marqueur sur le drone. La méthode de soustraction de l'arrière-plan ne permet pas d'avoir un autre objet en mouvement à par le drone. Plusieurs solutions sont alors possibles. Une première solution serait d'utiliser directement le capteur infrarouge afin de récupérer l'intensité lumineuse de l'environnement. On pourra utiliser un marqueur réfléchissant. Ce marqueur produira une intensité lumineuse beaucoup plus élevée que l'environnement. On pourra alors isoler le marqueur. Une autre méthode est d'utiliser la caméra couleur de la Kinect afin d'utiliser d'autre algorithme de détection d'objet (détection par la couleur, par la forme, etc.).

Un gros problème sur les drones est la chute de la tension de la batterie. La loi d'adaptation a permis de corriger ce problème. On peut cependant trouver d'autres solutions à ce problème. Un asservissement des vitesses des hélices peut être envisageable afin d'assurer en sortie du contrôleur que la force de portance désirée est obtenue.

Les paramètres du drone peuvent changer dans certaines conditions (transport d'une charge par exemple). Une estimation des paramètres permettrait alors d'assurer la stabilité du contrôleur de position et d'attitude.

L'utilisation de l'ordinateur embarqué Odroid suivi de la carte électronique Pixhawk peut s'avérer problématique pour l'implémentation du contrôleur sur le drone. L'Odroid peut s'avérer instable lorsque la carte est utilisée de manière prolongée. Le Pixhawk ne dispose pas de carte WI-FI afin de communiquer avec la base de contrôle. L'utilisation d'une autre carte électronique (le Navio par exemple) peut alors améliorer l'implémentation du contrôleur.

ANNEXE I

COEFFICIENTS DU FILTRE PASSE-BAS EN DISCRET

$$a_1 = -\frac{e^{-T_{ech}\,\omega_r\,\zeta}}{\sqrt{1-\zeta^2}}\left(\zeta\,\sin\left(T_{ech}\,\omega_r\sqrt{1-\zeta^2}\right) - e^{T_{ech}\,\omega_r\,\zeta}\sqrt{1-\zeta^2}\right. \qquad \text{(A I-1)}$$
$$\left. +\cos\left(T_{ech}\,\omega_r\sqrt{1-\zeta^2}\right)\sqrt{1-\zeta^2}\right)$$

$$a_2 = \frac{e^{-2\,T_{ech}\,\omega_r\,\zeta}}{\sqrt{1-\zeta^2}}\left(\sqrt{1-\zeta^2}+\zeta\,e^{T_{ech}\,\omega_r\,\zeta}\,\sin\left(T_{ech}\,\omega_r\sqrt{1-\zeta^2}\right)\right. \qquad \text{(A I-2)}$$
$$\left. -e^{T_{ech}\,\omega_r\,\zeta}\,\cos\left(T_{ech}\,\omega_r\sqrt{1-\zeta^2}\right)\sqrt{1-\zeta^2}\right)$$

$$a_3 = \frac{\omega_r\,e^{-T_{ech}\,\omega_r\,\zeta}\,\sin\left(T_{ech}\,\omega_r\sqrt{1-\zeta^2}\right)}{\sqrt{1-\zeta^2}} \qquad \text{(A I-3)}$$

$$a_4 = -\frac{\omega_r e^{-T_{ech}\omega_r\zeta}\,\sin\left(T_{ech}\,\omega_r\sqrt{1-\zeta^2}\right)}{\sqrt{1-\zeta^2}} \qquad \text{(A I-4)}$$

$$b_1 = -2\,e^{-T_{ech}\,\omega_r\,\zeta}\,\cos\left(T_{ech}\,\omega_r\sqrt{1-\zeta^2}\right) \qquad \text{(A I-5)}$$

$$b_2 = e^{-2\,T_{ech}\,\omega_r\,\zeta} \qquad \text{(A I-6)}$$

ANNEXE II

LEMME DE BARBALAT

Soit une fonction différentiable $\phi(t)$. Si $\phi(t)$ est uniformément continue pour tout $\forall t > t_0$, $t_0 \in R^*$ et $\int_{t_0}^{t} \phi(\tau) d\tau$ existe et est finie lorsque $t \to \infty$ alors $\lim_{t\to\infty} \phi(t) = 0$.

LISTE DE RÉFÉRENCES

Amiri, N., Ramirez-Serrano, A. & Davies, R. J. (2013). Integral backstepping control of an unconventional dual-fan unmanned aerial vehicle. *Journal of Intelligent & Robotic Systems*, 1-13.

Bandyopadhyay, B., Janardhanan, S. & Spurgeon, S. K. (2013). Adaptive Sliding Mode Control. Dans *Advances in sliding mode control : concept, theory and implementation* (vol. 440, pp. 21-37). Springer.

Benallegue, A., Mokhtari, A. & Fridman, L. (2008). High-order sliding-mode observer for a quadrotor UAV. *International journal of robust and nonlinear control*, *18*(4-5), 427-440.

Bouadi, H. & Tadjine, M. (2007). Nonlinear observer design and sliding mode control of four rotors helicopter. *Vorld Academy of Science, Engineering and Technology*, *25*, 225-229.

Bouwmans, T. (2014). Traditional and recent approaches in background modeling for foreground detection : An overview. *Computer science review*, *11*, 31–66.

Chan, A., Aguillon, J., Hill, D. & Lou, E. (2017). Precision and accuracy of consumer-grade motion tracking system for pedicle screw placement in pediatric spinal fusion surgery. *Medical Engineering & Physics*, 33-43.

Craig, J. J. (2005). Jacobians : velocitites and static forces. Dans *Introduction to robotics : mechanics and control* (vol. 3, pp. 138-139). Pearson Prentice Hall Upper Saddle River.

Das, A., Lewis, F. & Subbarao, K. (2009). Backstepping approach for controlling a quadrotor using lagrange form dynamics. *Journal of Intelligent and Robotic Systems*, *56*(1-2), 127-151.

Dassault Systèmes. (2017). Solidworks (Version 2017) [Logiciel]. Vélizy-Villacoublay, France : Dassault Systèmes.

Deters, R. W., Ananda, G. K. & Selig, M. S. (2014). Reynolds number effects on the performance of small-scale propellers. Dans *Proceedings of the 32nd AIAA Applied Aerodynamics Conference, Atlanta, GA, USA* (pp. 16-20).

Dougherty, E. R. & Lotufo, R. A. (2003). Binary opening and closing. Dans *Hands-on morphological image processing* (vol. 59, pp. 25-30). SPIE press.

Emlid. (2017a). Shop. Repéré le 15 juillet 2017 à https://emlid.com/shop/.

Emlid. (2017b). Navio2. Repéré le 15 juillet 2017 à https://emlid.com/navio/.

Erle. (2017a). Erle robotics. Repéré le 14 juillet 2017 à http://erlerobotics.com/blog/erle-brain-3/.

Erle. (2017b). Erle-brain. Repéré le 14 juillet 2017 à https://erlerobotics.gitbooks.io/.

Euston, M., Coote, P., Mahony, R., Kim, J. & Hamel, T. (2008, Septembre). *A complementary filter for attitude estimation of a fixed-wing UAV*. Communication présentée à IROS 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France (pp. 340-345).

Fallaha, C. J., Saad, M., Kanaan, H. Y. & Al-Haddad, K. (2011). Sliding-mode robot control with exponential reaching law. *IEEE Transactions on Industrial Electronics*, *58*(2), 600-610.

González, I., Salazar, S. & Lozano, R. (2014). Chattering-free sliding mode altitude control for a quad-rotor aircraft : Real-time application. *Journal of Intelligent & Robotic Systems*, *73*(1-4), 137-155.

Groizeleau, V. (2014). Aero surveillance. Repéré le 8 aout 2017 à https://www.meretmarine.com/fr/content/aero-surveillance-devoile-son-nouveau-drone-voilure-tournante.

Grover, R. & Hwang, P. Y. (2012). Kalman filtering and applications. Dans *Introduction to random signals and applied Kalman filtering* (éd. 4, pp. 139-173). New York : Willey.

Harris, C. M. & Piersol, A. G. (2002). Vibration of a resiliently supported rigid body. Dans Harris, C. M. & Piersol, A. G. (Éds.), *Harris' shock and vibration handbook* (éd. 5, pp. 3.18-3.20). New York, NY : McGraw-Hill.

Hartley, R. & Zisserman, A. (2003). Camera models. Dans *Multiple View Geometry in Computer Vision* (éd. 2, pp. 159-173). Cambridge university press.

Herrera, D., Kannala, J. & Heikkilä, J. (2011). Accurate and practical calibration of a depth and color camera pair. *International Conference on Computer analysis of images and patterns*, pp. 437-445.

Higgins, W. T. (1975). A comparison of complementary and Kalman filtering. *IEEE Transactions on Aerospace and Electronic Systems*, *11*(3), 321-325.

Hoenig, W., Milanes, C., Scaria, L., Phan, T., Bolas, M. & Ayanian, N. (2015). Mixed reality for robotics. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5382-5387.

Hoffmann, G. M., Huang, H., Waslander, S. L. & Tomlin, C. J. (2007). Quadrotor helicopter flight dynamics and control : Theory and experiment. Dans *AIAA Guidance, Navigation and Control Conference and Exhibit* (vol. 2, pp. 4-7).

Hughes, A. (2006a). Synchronous and brushless D.C motors. Dans Newnes (Éd.), *Electric Motors and Drives* (éd. 3, pp. 357-360). Oxford : Elsevier.

Hughes, A. (2006b). Conventional DC motors. Dans Newnes (Éd.), *Electric Motors and Drives* (éd. 3, pp. 82-106). Oxford : Elsevier.

Jafari, H., Zareh, M., Roshanian, J. & Nikkhah, A. (2010). An optimal guidance law applied to quadrotor using LQR method. *Transactions of the Japan Society for Aeronautical and Space Sciences*, *53*(179), 32-39.

Jia, Y.-B. (2016). Quaternions and rotations. *Com S*, *477*(577), 15-36.

Kalman, R. E. et al. (1960). A new approach to linear filtering and prediction problems. *Journal of basic engineering*, *82*(1), 35-45.

Kanellakopoulos, I., Kokotovic, P. V. & Morse, A. S. (1991). Systematic design of adaptive controllers for feedback linearizable systems. *American Control Conference, 1991*, pp. 649-654.

Koehl, A. (2012). *Modélisation, observation et commande d'un drone miniature à birotor coaxial*. (Thèse de doctorat, Université Henri Poincaré-Nancy I, Nancy, France).

Landry, B. (2015). *Planning and control for quadrotor flight through cluttered environments*. (Thèse de doctorat, Massachusetts Institute of Technology, Cambridge, MA).

Lee, D., Kim, H. J. & Sastry, S. (2009). Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter. *International Journal of control, Automation and systems*, *7*(3), 419-428.

Luis, C. & Ny, J. L. (2016). *Design of a trajectory tracking controller for a nanoquadcopter*. Montréal : Polytechnique Montréal.

Madgwick, S. O., Harrison, A. J. & Vaidyanathan, R. (2011). Estimation of imu and marg orientation using a gradient descent algorithm. *2011 IEEE International Conference on Rehabilitation Robotics (ICORR)*, pp. 1-7.

Mahony, R., Hamel, T. & Pflimlin, J.-M. (2005). Complementary filter design on the special orthogonal group SO (3). *2005 44th IEEE Conference on Decision and Control and European Control Conference Decision and Control. CDC-ECC'05*, pp. 1477-1484.

Marins, J. L., Yun, X., Bachmann, E. R., McGhee, R. B. & Zyda, M. J. (2001). An extended Kalman filter for quaternion-based orientation estimation using MARG sensors. *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4, 2003-2011.

MathWorks. (2017). Matlab (Version R2016a) [Logiciel]. Natick, MA : MathWorks.

Mavlink. (2015). MaLink common message set [Documentation]. Repéré le 25 juillet 2017 à http://mavlink.org/messages/common.

McCabe, B., Hamledari, H., Shahi, A., Zangeneh, P. & Azar, E. R. (2017). Roles, benefits, and bhallenges of using UAVs for indoor smart construction applications. Dans *Computing in Civil Engineering 2017* (pp. 349-357).

Mer et Marine. (2014). RQ-4 Global Hawk. Repéré le 8 aout 2017 à https://www.meretmarine.com/fr/content/aero-surveillance-devoile-son-nouveau-drone-voilure-tournante.

Nägeli, T., Alonso-Mora, J., Domahidi, A., Rus, D. & Hilliges, O. (2017). Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization. *IEEE Robotics and Automation Letters*, *2*(3), 1696-1703.

Pixhawk. (2017). Pixhawk home. Repéré le 15 juillet 2017 à https://pixhawk.org/.

Pulver, A., Wei, R. & Mann, C. (2016). Locating AED enabled medical drones to enhance cardiac arrest response times. *Prehospital Emergency Care*, 20(3), 378-389.

Raffo, G. V., Ortega, M. G. & Rubio, F. R. (2008). Backstepping/nonlinear H∞ control for path tracking of a quadrotor unmanned aerial vehicle. *American Control Conferences*, pp. 3356-3361.

Robin, T. (2017). *Sliding mode controller for a quadrotor*. (Mémoire de maîtrise, University of Victoria, Inde).

RobotShop. (2017). Crazyflie 2.0 mini quadcopter. Repéré le 15 juillet 2017 à http://www.robotshop.com/uk/crazyflie-20-mini-quadcopter.html.

Sarbolandi, H., Lefloch, D. & Kolb, A. (2015). Kinect range sensing : Structured-light versus time-of-flight kinect. *Computer Vision and Image Understanding*, *139*, 1-20.

Simon, D. (2006). Alternate Kalman filter formulations. Dans *Optimal state estimation : Kalman, H∞, and nonlinear approachess* (pp. 151-177). John Wiley & Sons.

Sobral, A. & Vacavant, A. (2014). A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, 122, 4-21.

Subramanian, G. P. (2015). *Nonlinear control strategies for quadrotors and cubesats*. (Thèse de doctorat, University of Illinois, Urbana, Illinois).

Sung, K. (2003). Fuzzy logic based closed-loop strapdown attitude system for unmanned aerial vehicle. *Sensors and Actuators A : Physical*, *107*(1), 109-118.

Windolf, M., Götzen, N. & Morlock, M. (2008). Systematic accuracy and precision analysis of video motion capturing systems exemplified on the Vicon-460 system. *Journal of biomechanics*, *41*(12), 2776-2780.

Woodman, O. J. (2007). *An introduction to inertial navigation*. Cambridge : University of Cambridge, Computer Laboratory.

Yang, H. C., Sababha, B., Acar, C. & Rawashdeh, O. (2010). Rapid Prototyping of quadrotor controllers using Matlab RTW and dsPics. *Proceedings of the AIAA Infotech@ Aerospace Conference, Georgia, USA*.

Yang, S., Scherer, S. A., Schauwecker, K. & Zell, A. (2013). Onboard monocular vision for landing of an MAV on a landing site specified by a single reference image. *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 318-325.

Yao, N., Anaya, E., Tao, Q., Cho, S., Zheng, H. & Zhang, F. (2017). Monocular vision-based human following on miniature robotic blimp. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3244-3249.

Yaou, Z., Wansheng, Z., Tiansheng, L. & Jingsong, L. (2013). The attitude control of the four-rotor unmanned helicopter based on feedback linearization control. *WSEAS Transactions on Systems*, (4), 229-239.

Zuo, Z. (2013). Adaptive trajectory tracking control design with command filtered compensation for a quadrotor. *Journal of Vibration and Control*, *19*(1), 94-108.